

**Studies in Layout-driven Routing, Thermal Problems  
and  
Delay Fault Classification for VLSI Physical Design**

*Dissertation submitted for the degree*

*of*

*Doctor of Philosophy (Engineering)*

*of*

*Jadavpur University*



**SUBHASHIS MAJUMDER**

Dept. of Computer Science & Engineering

Jadavpur University

Kolkata -700 032, INDIA

October, 2005

**Studies in Layout-driven Routing, Thermal Problems  
and  
Delay Fault Classification for VLSI Physical Design**

*Dissertation submitted for the degree*

*of*

*Doctor of Philosophy (Engineering)*

*of*

*Jadavpur University*



Subhashis Majumder

Dept. of Computer Science & Engineering

Jadavpur University

Kolkata -700 032, INDIA

October, 2005

*To my parents*

*Smt. Minati Majumder*

*and*

*Sri Kalyan Kumar Majumder*

## Acknowledgment

I express my deepest gratitude and indebtedness to my supervisors Prof. Bhargab B. Bhattacharya of the Indian Statistical Institute, Kolkata and Prof. Debesh K. Das of the Jadavpur University for their support. Prof. Bhattacharya has been my friend, philosopher and guide in the truest sense.

My deep gratitude also goes to my parents, to my wife and to little Ditsa who have sacrificed a lot to see this day. In fact, I owe this thesis to my father, it is rather his thesis than mine.

I am specially grateful to Prof. S. C. Nandy and Prof. S. Sur-Kolay for their timely interventions and stimulating discussions throughout my research career in ISI, Kolkata. I also thank Prof. B. P. Sinha and all other members of ACM Unit of ISI for maintaining such an excellent research environment.

I sincerely acknowledge the support I got from my Director Mr. S. Bose and all other colleagues of IIIT, Kolkata that ultimately helped me to finish my work. I would like to thank all my students especially S. M. A. Jafri, V. Bansal and V. Chowdhary, it is for them I enjoy academics. Thanks are also due to my co-authors Swarup Das and Swarup Bhunia.

I acknowledge the efforts of the doctors Dr. A. Bandopadhyay, Dr. S. Bhaduri, Dr. D. Chowdhury, Dr. S. Dasgupta, Dr. A. K. Pal and Dr. D. Pahari for their sincere medical efforts behind my father in the last few years.

Finally, I thank Prof. D. Mukherjee and Prof. A. K. Bandopadhyay of Jadavpur University, Prof. M. L. Bushnell and Prof. C. Rose of Rutgers University and Prof. Vishwani D. Agrawal of Auburn University who did not allow me to think that not doing a Ph.D is an option. I wish with all their blessings this small effort becomes the beginning of a new chapter in my life.

October 05, 2005

Subhashis Majumder

International Institute of Information Technology

Kolkata – 700091, India

# Abstract

With ever increasing performance demand, physical design has turned out to be one of the most important steps in the VLSI design cycle. Traditionally it was divided into several classical steps, but to combat the challenges posed by shrinking device dimensions and increasing interconnect delay, several new steps have been integrated with the physical design cycle. Various aspects of VLSI physical design, both classical and modern, have been addressed in this thesis.

Three new areas of physical design have been studied, namely, (i) layout-driven floorplan partitioning and routing, (ii) geometric aspects of thermal problems in hot chips and related management issues, and (iii) analysis of path-delay faults in terms of classical stuck-at faults.

In layout-driven floorplan partitioning, a new area-partitioning scheme has been proposed to facilitate global routing and repeater placement issues. Several theoretical results have been proved and finally a max-flow-min-cut based heuristic has been proposed to hierarchically partition the floorplan using monotone staircases. Experimental results on benchmark floorplans are also presented. Next, a new routing technique called topological routing has been proposed that tries to route all the nets simultaneously, unlike the traditional approach of routing net-by-net. This novel method has produced very encouraging results in routing randomly generated testcases of moderate to large size. Finally, a new layout problem, that of minimizing the number of intersections among multiple Steiner trees is studied.

In the area of thermal analysis, first a simple geometric model is proposed to evaluate the cumulative thermal power at any point of the chip floor, followed by a discrete version of that model. Then a simulation based approach is presented that identifies the hot spots and zones on a chip floor using the above models. Next, to have an overall saving in simulation time, a hybrid methodology is proposed that first uses a Voronoi diagram based technique, and then uses simulation only in some focussed zones of the chip. Last but not the least, a new concept of density of points has been introduced. Several geometric results that facilitate the identification of regions with maximum/minimum density have been proved. The concept of density has then been compared with an existing concept of discrepancy to show that there may be some distinct advantage in applying the concept of density over discrepancy in thermal analysis of hot spots in a chip.

In this thesis, a very detailed theory is given how to model a path-delay fault in terms of an equivalent stuck-at fault(s) in a modified circuit derived from the original circuit. Many problems of physical design, e.g. power droop management and repeater placement may need simulation studies via delay fault model. A uniform methodology is presented for finding an equivalent stuck-at fault for several existing classifications. As the stuck-at fault domain is much better studied compared to the delay fault domain, this work might help in analyzing delay faults better, and in turn may offer an efficient tool for timing analysis in high performance circuits.

**Keywords:** Floorplanning, global routing, NP-completeness, heuristics, network flow, Steiner tree, linear programming, hot spot, Voronoi diagram, discrepancy, stuck-at fault, delay fault.

# List of Publications of the Author Related to this Thesis

1. S. Majumder, B. B. Bhattacharya and S. C. Nandy, "Partitioning VLSI floorplans by staircase channels for global routing", *Proc. of the 11th Int. Conf. on VLSI Design*, pp. 59-64, 1998.
2. S. Majumder, V. D. Agrawal and M. L. Bushnell, "On delay untestable paths and stuck-fault redundancy", *Proc. of the 16th VLSI Test Symposium*, pp. 194-199, 1998.
3. S. Majumder, B. B. Bhattacharya, V. D. Agrawal and M. L. Bushnell, "A complete characterization of path delay faults through stuck-at faults", *Proc. of the 12th Int. Conf. on VLSI Design*, pp. 492-497, 1999.
4. S. Bhunia, S. Majumder, S. Sur-Kolay, A. Sircar and B. B. Bhattacharya, "Topological routing amidst polygonal obstacles", *Proc. of the 13th Int. Conf. on VLSI Design*, pp. 274-279, 2000.
5. S. Majumder, S. Sur-Kolay, B. B. Bhattacharya and S. C. Nandy, "Area (Number) balanced hierarchy of staircase channels with minimum crossing nets", *Proc. of ISCAS 2001*, V, pp. 395-398, 2001.

6. S. Majumder, S. C. Nandy and B. B. Bhattacharya, "On finding a staircase channel with minimum crossing nets in a VLSI floorplan", *Journal of Circuits, Systems and Computers*, Vol. 13(5), pp. 1019-1038, 2004.
7. S. Majumder, B. B. Bhattacharya, V. D. Agrawal and M. L. Bushnell, "A new classification of path-delay fault testability in terms of stuck-at faults", *Journal of Computer Science and Technology*, Vol. 19(6), pp. 955-964, 2004.
8. S. Majumder, S. Sur-Kolay, S. C. Nandy, B. B. Bhattacharya and B. Chakraborty, "Hot spots and zones in a chip: A geometricians view", *Proc. of the 18th Int. Conference on VLSI Design*, pp. 691-696, 2005.
9. S. Majumder and B. B. Bhattacharya, "Density or discrepancy? A VLSI designer's dilemma in hot spot analysis", *Proc. of the Canadian Conference on Computational Geometry*, pp. 167-170, 2005.
10. S. Majumder, S. Sur-Kolay, B. B. Bhattacharya and S. Das, "Hierarchical partitioning of VLSI floorplans by staircases", *submitted for journal publication*, 2005.
11. S. Majumder, B. B. Bhattacharya and Syed M. A. Jafri, "Reducing crossing number of multi-color rectilinear steiner trees using monochromatic partitioning", *submitted for publication*, 2005.
12. S. Majumder, S. C. Nandy and B. B. Bhattacharya, "Separating multi-color points on a plane with fewest axis-parallel lines", *Tech. Report ISI/ACMU/VLSI - 1*, Kolkata, 2005.

13. S. Majumder, S. Sur-Kolay, S. Bhunia and B. B. Bhattacharya, "Topological routing for a group of nets", *Tech. Report ISI/ACMU/VLSI - 2*, Kolkata, 2005.
14. S. Majumder and B. B. Bhattacharya, "Solving thermal problems of hot chips using Voronoi diagrams", *Proc. of the 19th Int. Conference on VLSI Design*, Jan 2006 (to appear).

# List of Presentations in International Conferences

1. “Partitioning VLSI floorplans by staircase channels for global routing”,  
*Presented at the 11th Int. Conf. on VLSI Design*, Chennai, India, 1998.
2. “On delay untestable paths and stuck-fault redundancy”, *Presented at  
the 16th VLSI Test Symposium*, Princeton, U.S.A., 1998.
3. “A complete characterization of path delay faults through stuck-at faults”,  
*Presented at the 12th Int. Conf. on VLSI Design*, Goa, India, 1999.
4. “Topological routing amidst polygonal obstacles”, *Presented at the 13th  
Int. Conf. on VLSI Design*, Kolkata, India, 2000.
5. “Area (Number) balanced hierarchy of staircase channels with minimum  
crossing nets”, *Presented at ISCAS 2001*, Sydney, Australia, 2001.
6. “Hot spots and zones in a chip: A geometricians view”, *Presented at the  
18th Int. Conference on VLSI Design*, Kolkata, India, 2005.
7. “Density or discrepancy? A VLSI designer’s dilemma in hot spot analy-  
sis”, *Presented at the Canadian Conference on Computational Geometry*,  
Windsor, Canada, 2005.

# Contents

<b>Abstract</b>	<b>i</b>
<b>List of Publications of the Author Related to this Thesis</b>	<b>iii</b>
<b>List of Presentations in International Conferences</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 The VLSI design cycle . . . . .	2
1.2 The physical design cycle . . . . .	3
1.3 Scope of the thesis . . . . .	4
1.3.1 Layout-driven partitioning and routing . . . . .	4
1.3.2 Thermal problem of hot chips . . . . .	6
1.3.3 Analysis and classification of path-delay faults . . . . .	7
1.4 Organization of the thesis . . . . .	9
<b>2 Floorplan-driven Partitioning</b>	<b>10</b>
2.1 Introduction . . . . .	10
2.2 Floorplan partitioning problem . . . . .	11
2.3 Motivation for the <i>MSC</i> problem . . . . .	13

2.3.1	Repeater placement problem . . . . .	14
2.3.2	Channel/global routing . . . . .	15
2.4	Graph-theoretic preliminaries . . . . .	16
2.4.1	Monotone <i>st</i> -cut and staircase partitioning . . . . .	18
2.5	Two-terminal net problem . . . . .	20
2.5.1	Minimizing the number of crossing nets . . . . .	21
2.5.2	Algorithm . . . . .	24
2.5.3	Complexity analysis . . . . .	25
2.6	Multi-terminal net problem . . . . .	25
2.6.1	Some geometric concepts . . . . .	26
2.6.2	Formulation . . . . .	29
2.6.3	Algorithm and its complexity analysis . . . . .	35
2.7	Hierarchical partitioning with staircases . . . . .	35
2.8	Generation of a balanced hierarchy . . . . .	37
2.9	Problem formulation . . . . .	39
2.10	Area-balanced monotone staircase bipartitioning . . . . .	40
2.11	Heuristic for mixed optimization . . . . .	44
2.11.1	Flow based balanced bipartitioning . . . . .	44
2.11.2	Overview of our heuristic . . . . .	45
2.12	Experimental results . . . . .	47
2.13	Conclusion . . . . .	53
<b>3</b>	<b>Topological Routing</b>	<b>54</b>
3.1	Introduction . . . . .	54

---

3.2	Motivation . . . . .	54
3.3	Trapezoidal decomposition . . . . .	56
3.4	Routing algorithm . . . . .	63
3.4.1	Overview . . . . .	63
3.4.2	Algorithm for routing by traversal of dual graph . . . . .	64
3.5	Implementation of our algorithm . . . . .	70
3.5.1	Generation of a polygon within a given rectangle . . . . .	70
3.5.2	Random generation of terminals on polygons . . . . .	71
3.5.3	Experimental results . . . . .	72
3.6	Concluding remarks . . . . .	72
<b>4</b>	<b>Reducing Intersections for Rectilinear Steiner Trees</b>	<b>75</b>
4.1	Introduction . . . . .	75
4.2	Background of the problem . . . . .	76
4.3	Monochromatic partitioning . . . . .	78
4.3.1	Prior work . . . . .	78
4.3.2	Some important observations . . . . .	80
4.3.3	NP-completeness results . . . . .	82
4.3.4	A fast heuristic . . . . .	87
4.3.5	Performance of our greedy heuristic . . . . .	93
4.4	Solution methodology for the intersection problem . . . . .	94
4.4.1	Working principle of the heuristic $H_s$ . . . . .	96
4.5	Experimental results . . . . .	97
4.6	Conclusion . . . . .	100

<b>5</b>	<b>Geometric Analysis of Hot Spots</b>	<b>101</b>
5.1	Introduction . . . . .	101
5.2	Models for thermal analysis . . . . .	102
5.3	Time-invariant heat sources . . . . .	103
5.3.1	Continuous spatial domain . . . . .	103
5.3.2	Discrete spatial domain . . . . .	107
5.3.3	Window model in discrete domain . . . . .	111
5.4	Time-varying heat sources . . . . .	112
5.5	Experimental results . . . . .	113
5.6	A hybrid approach . . . . .	116
5.6.1	Approximate prediction of hot zones . . . . .	116
5.6.2	Local simulation on critical Voronoi cells . . . . .	122
5.7	Dispersing the hot spots . . . . .	124
5.7.1	Defining the safe zone . . . . .	126
5.8	Concluding remarks . . . . .	127
<b>6</b>	<b>Density vs. Discrepancy of Points</b>	<b>128</b>
6.1	Introduction and prior work . . . . .	128
6.2	Problem formulation . . . . .	129
6.3	Problem $P_{min}$ . . . . .	130
6.3.1	Algorithm . . . . .	131
6.3.2	Density in general case . . . . .	134
6.4	Problem $P_{max}$ . . . . .	134
6.5	Density vs. discrepancy . . . . .	136

---

6.6	Conclusion and future works . . . . .	138
<b>7</b>	<b>Classification of Path-delay Faults Using Stuck-at Faults</b>	<b>140</b>
7.1	Introduction . . . . .	140
7.2	Circuit unfolding . . . . .	142
7.3	Stuck-at faults in unfolded circuit and path classifications . . . .	145
7.4	Conclusion . . . . .	161
<b>8</b>	<b>Conclusions and Future Works</b>	<b>162</b>
	<b>Bibliography</b>	<b>165</b>

# List of Figures

2.1	(a) A staircase channel in a layout and (b) A staircase in the corresponding floorplan . . . . .	12
2.2	A typical example of repeater placement . . . . .	13
2.3	(a) A floorplan, (b) A non-monotone st-cut, and (c) A monotone st-cut of its adjacency digraph . . . . .	17
2.4	Four types of L-paths . . . . .	20
2.5	Illustration of weighted arcs in the graph $G_{aug}$ . . . . .	22
2.6	Stepwise construction of $G_{aug}$ (capacity of dotted edges = $\infty$ ) . . . . .	23
2.7	UPSTAIR, DOWNSTAIR and INTZONE for net $c_\alpha$ . . . . .	28
2.8	Overlap of INTZONES for two nets . . . . .	28
2.9	Stepwise construction of $G_{aug}^M$ (capacity of dotted edges = $\infty$ ) . . . . .	31
2.10	Hierarchy of monotone staircases in a floorplan . . . . .	38
2.11	Floorplans for proof of NP-completeness . . . . .	41
2.12	Computation of $\delta$ . . . . .	43
2.13	Effect of $\gamma$ on minimum and maximum values of cost over all levels of number and area-balance hierarchy of ms-cuts . . . . .	48
2.14	First level statistics . . . . .	50
2.15	Variation of number of levels and maximum cutsize . . . . .	52

---

3.1	Main steps of the algorithm . . . . .	56
3.2	Generation of trapeziums in $F$ . . . . .	58
3.3	Types of end-points for trapezium edges . . . . .	59
3.4	Matching of top and bottom edges for a trapezium . . . . .	60
3.5	Computation of vertex and edge constraints . . . . .	61
3.6	Dual graph for a typical example . . . . .	62
3.7	Original and inherited terminal lists for net $a$ . . . . .	64
3.8	An example of broadcasting and routing in the forward pass . . . . .	65
3.9	An example of broadcasting and routing in the backward pass . . . . .	68
4.1	A counterexample for monochromatic bands . . . . .	80
4.2	An example graph for the proof of NP-completeness . . . . .	82
4.3	Reduction from vertex cover . . . . .	83
4.4	Sample run of our greedy heuristic . . . . .	91
4.5	A typical example showing the steps of the heuristic $H_s$ . . . . .	95
5.1	Unit circle model of heat received at point $T$ . . . . .	104
5.2	Special cases of the unit circle model . . . . .	105
5.3	Thermal pattern within an octagon . . . . .	106
5.4	Typical thermal distribution for 10 sources . . . . .	107
5.5	Propagation model for discrete domain . . . . .	108
5.6	Propagation coefficients in the grid . . . . .	109
5.7	Moving window over the discrete grid . . . . .	112
5.8	Weighted Voronoi diagram for 11 heat sources, based on raw strengths	119
5.9	Superposition of simulation result and Voronoi diagram of 11 weighted heat sources . . . . .	121

5.10	Ordinary Voronoi diagram for 11 heat sources of uniform weight . . .	122
5.11	Superposition of simulation result and Voronoi diagram of 11 uniformly weighted heat sources . . . . .	123
5.12	Moving source $D$ in a portion of the floor that has crossed the thermal threshold . . . . .	125
5.13	Safe zone of movement for source $H$ . . . . .	126
6.1	Smallest rectangle containing $k$ points . . . . .	130
6.2	Rectangles with corners in opposite quadrants . . . . .	133
6.3	Density in a non-isothetic scenario . . . . .	135
6.4	Largest rectangle containing $k$ points . . . . .	136
6.5	Local discrepancy of a set of points . . . . .	137
7.1	(a) Original circuit $C$ (b) Unfolded circuit $U$ . . . . .	143
7.2	Classifications of path-delay faults . . . . .	145
7.3	Singly-testable paths . . . . .	147
7.4	Multiply-testable paths . . . . .	149
7.5	(a) Original circuit (b) Unfolded circuit (c) Limitations of a non-robust test . . . . .	151
7.6	A functionally sensitized ST-dependent path . . . . .	154
7.7	A typical circuit and its two-input equivalent . . . . .	156
7.8	A typical example of FS path . . . . .	159
7.9	An example for Case 6 . . . . .	160

# List of Tables

2.1	Details of MCNC floorplan benchmarks . . . . .	46
2.2	Effect of $\gamma$ on number-balanced hierarchy for <i>ami33</i> . . . . .	47
2.3	Effect of $\gamma$ on area-balanced hierarchy for <i>ami33</i> . . . . .	49
2.4	Variation of number of levels and maximum cutsize with $\gamma$ . . . . .	51
3.1	Experimental results . . . . .	73
4.1	Separation of bi-color points . . . . .	92
4.2	Separation of multi-color points . . . . .	93
4.3	Performance of heuristic for two colors . . . . .	98
4.4	Performance of heuristic for more than two colors . . . . .	99
5.1	Results for the continuous domain . . . . .	113
5.2	Results for the discrete domain . . . . .	114
5.3	Results for window model . . . . .	115
5.4	Power window results with varying threshold . . . . .	116
5.5	Thermal data for 11 weighted sources . . . . .	120
5.6	Thermal data for 11 unweighted sources . . . . .	124
7.1	Possible transitions . . . . .	157

# Chapter 1

## Introduction

The overwhelming impact of integrated circuits (IC) on electronic industry can be compared only with the revolutionary change that the press had brought to the printing industry in the history of human civilization. In a way, both the events have achieved similar goals by enabling dissemination of knowledge to the common man at an affordable price. Needless to say, the domain of integrated circuits encompasses various other walks of life, though their contribution towards enhancing knowledge in this era of information technology has been the greatest reward for the human race. Since its invention by Jack Kilby of the Texas Instruments in 1959, the IC has always produced wonders. However, the rapid progress in speed, size and cost for any electronic system started to become conspicuous, with the advent of VLSI (Very Large Scale Integration). The performance/cost ratio of the VLSI chips has been increasing at an approximate rate of  $10^3$  for every ten years for the last few decades. Such a phenomenal improvement in performance coupled with low per-unit cost has imparted an all pervasive effect on the modern society. In fact, VLSI chips have become indispensable in almost all branches of engineering and science including telecommunication, factory automation, space research, bioengineering, bio-informatics, aeronautics, weather forecasting, as well as in computing,

to name a few. The cycle becomes complete when today's computers powered with VLSI chips strive to make it possible to build tomorrow's chips and the exponential nature of improvement has led to the emergence of a new design paradigm known as GSI (Giga Scale Integration).

### 1.1 The VLSI design cycle

VLSI Design Cycle has evolved into a very complex process with a large number of steps to be carried out depending upon the design style [113]. However, the different steps can be divided into three broad categories – design and verification of the chip, fabrication of the chip, and finally testing of the fabricated chip. VLSI is basically the art and science of putting millions of electronic components, most commonly on a silicon base called a substrate by layering several materials in a well-defined fashion. In fact, the job of a VLSI designer is to transform a circuit description into a geometric description called the layout. A layout consists of a set of planar geometric shapes in several layers. Once created, the layout is checked to ensure that it meets all the design requirements. The layout is generally described by a set of design files, from which optical patterns called *masks* are generated. During fabrication, these masks are used to create patterns on silicon by using a sequence of photolithographic steps. Also for fabrication, component formation requires very fine details about geometric patterns and separation between them (design rules). The process of converting the specification of an electrical circuit into a layout is called the *physical design process*.

The VLSI design cycle starts with a specification for a VLSI chip, goes through a series of steps and finally produces a packaged chip. The steps followed by a typical design cycle are System Specification, Functional Design, Logic Design, Circuit Design, Verification, Physical Design, Fabrication,

Packaging and Testing. However, with shrinking feature size there have been some new trends in the design cycle. The most important of them is to combat increasing interconnect area and delay, and thermal management. The interconnect design challenges, thermal issues, along with the complexity of handling an increasing number of available metal layers have made physical design considerations to enter into the design cycle at a much earlier phase. This helps in better optimization of area and timing parameters. In order to reduce design iterations as well as to achieve high performance, current logic synthesis tools interact constantly with layout synthesis as well. Also to test whether the fabricated chip indeed achieves the specified performance criteria, testing of timing faults (e.g., delay or transition faults) have become very important [15]. High performance chips like microprocessors have to go through the delay/transition fault testing phase together with the traditional ones like stuck-at fault and IDDQ testing. In short, with everlasting performance demand, physical design of integrated circuits will remain as one of the most important steps in the VLSI/GSI design cycle.

## 1.2 The physical design cycle

The physical design step itself is a long process being accomplished in several stages, and hence can be called the physical design cycle. Typically, the input to this cycle is a netlist and the output is an optimized layout of the circuit. The various stages of physical design are partitioning, floorplanning and placement, routing – global and detailed, compaction, extraction, and verification. Physical design like the full VLSI design cycle is iterative in nature and many steps may have to be repeated to generate the final layout [113, 107]. It may also be needed to go back and forth through the stages in order to generate a layout satisfying the target specifications. As for example, a placement

scheme of poor quality may not be fully compensated by a high quality routing phase at a subsequent stage, thus requiring a thorough overhaul in the earlier placement phase in order to make the design properly routable. Though speed and area had been the traditional design goals, with the advent of hand-held devices, power requirement has become one of the important design issues. Consequently, the heat dissipation problem, i.e., cooling down the hot chips, turns out to be a challenging one [56, 105, 51]. Ever increasing demand for performance and shrinking device dimensions are responsible for excessive heating of a chip. Thus, thermal management has now become an integral part of the physical design cycle.

### 1.3 Scope of the thesis

In this thesis, we investigate primarily on three new problems of physical design, namely (i) layout-driven floorplan partitioning and routing, (ii) geometric aspects of thermal problems in hot chips and related management issues, and (iii) analysis of path delay faults in terms of classical stuck-at faults. The last problem has also become important to a physical designer as several issues involving power grid design [97], interconnect layout and buffer/repeater insertion [110, 106, 55], may strongly influence the timing behavior of the circuit.

#### 1.3.1 Layout-driven partitioning and routing

To facilitate global routing and repeater placement, we address three new problems: (i) floorplan-driven partitioning, (ii) topological routing, and (iii) reducing intersections among multiple Steiner trees.

##### (i) Floorplan-driven partitioning

Floorplanning is becoming a very important aspect in VLSI physical design [129,

38] as circuit size is growing very rapidly and hierarchical design with IP blocks is being widely used to reduce the design complexity [1]. Integrating the various steps of physical design like partitioning, floorplanning, and placement is one of the latest trends in both digital and analog domain [2, 10, 68].

Given a VLSI floorplan, we introduce a new area-partitioning scheme based on monotone staircase channels (empty regions on silicon). The problem is to partition the floorplan hierarchically using monotonically increasing staircase cuts (MSC), so that the number of distinct nets crossing the cut is minimized, at each stage of the hierarchy [82, 85, 81, 86, 91]. A novel graph-based polynomial-time algorithm has been proposed to solve this problem. However, the two partitions obtained by this scheme are sometimes quite unbalanced in size. To take care of that, the problem has been further refined to create the partitions within a specified balance factor acceptable to the user. Several results have been proved regarding the computational hardness of the latter problem, and finally a max-flow-min-cut based heuristic has been proposed to solve it. The solution of the MSC problem has been shown to be useful for global routing and repeater placement in deep submicron IC chips.

### **(ii) Topological routing**

We investigate on a new routing scheme called topological routing that tries to route all the nets simultaneously. In the traditional approach, routing is done net-by-net. The disadvantage of this method is twofold: first, it is hard to determine an appropriate order; second, the ordering may cause a former net to hinder the routability of a latter net. Further, it is very difficult to control the congestion of routing areas, which renders the repeater placement problem very hard to handle. However, in topological routing, the routing order does not play any substantial role in determining the routability of any net. The novel method proposed in this thesis has produced very encouraging results in routing randomly generated testcases of moderate to large size [12, 93].

### (iii) Reducing intersections among multiple Steiner trees

We consider a new problem in global routing that is modeled as minimizing the number of intersections among multiple Steiner trees, and an efficient heuristic is designed [90, 92]. It may in turn reduce the number of vias required by the nets to change metal layers. Reducing vias is important as they are expensive and too much use of vias makes the circuit slow as well as reduces the yield of the chip.

### 1.3.2 Thermal problem of hot chips

The power density of today's high performance IC chips is too large, and as a result, identification of hot spots and subsequently design of cooling mechanisms including thermal vias and heat sinks has become very important [24, 25, 127, 105, 52, 51]. For analysis of hot spots, heat diffusion equations are usually employed to calculate their effects. In this thesis, we have described a completely new model to analyze the behavior of hot spots based on a purely geometric approach.

#### (i) Analysis of hot spots

A geometric model is proposed to evaluate the cumulative thermal power at any point of the chip floor. This is followed by a discrete version of the model that works in the grid domain. Then a simulation based approach is presented that identifies the hot spots and zones on a chip floor using the above models. Contrary to natural intuition, it is found that some non-source points on the chip floor can become more heated than heat-generating source points due to cumulative heating effect of nearby sources [88]. To improve over the time-consuming simulation based approach, a Voronoi diagram [9] based technique is further proposed that helps in quickly identifying the hot zones. To get a more accurate analysis, the simulation based approach can then be administered

in and around these hot zones. The final outcome of this approach is an overall saving in simulation time [94]. Further, we describe a procedure to dilate the hot zones by shifting some of the heat generating sources away from each other in order to make the hot zones cool down below the critical level. The movements are made within some restrictions so that the topology of the Voronoi diagram constructed around the point heat sources does not change. This technique while dilating the heat sources, will not change the floorplan adjacency relations significantly.

### (ii) Density vs. discrepancy

We introduce a new concept of *density* in an ensemble of points on a 2D floor, and prove several geometric results that facilitate the identification of regions with maximum/minimum density of source points on the chip floor. In the light of these results, the algorithms required to identify the most dense (hot) zone, as well as the most rarefied (cool) zone turn out to be quite simple. The concept of density has then been compared with an existing concept of discrepancy [6] and it is shown that there appears to be some distinct advantage in applying the concept of density over discrepancy in thermal analysis of hot spots in a chip [89].

### 1.3.3 Analysis and classification of path-delay faults

As mentioned earlier, many physical design issues, for example, power grid layout and interconnect/repeater management, may strongly influence the timing behavior of the chip. In a complex chip, when several cells that are drawing power ( $V_{dd}$ ) through a common power via from the power grid, switch simultaneously, a voltage drop (commonly known as power droop) may occur at the corresponding via. This may cause a slow-to-rise transition at the outputs of victim cells driven by the via [97]. Similarly, interconnect layout strategy

## Introduction

---

is affected by congestion as well as availability of room on the silicon floor to place the required number of repeaters in order to reduce delay and to minimize skew [110, 55, 26]. Correctness of such layout designs can only be verified by running/simulating delay tests [73]. Thus modeling, analysis, and testing of transition/delay faults [63, 77, 125], are extremely important to a chip designer.

The analysis of path-delay faults has two primary targets. The first is to directly identify the delay faults, i.e. slow paths and reject a chip that does not satisfy the target timing requirements. The second one is to identify the false paths, i.e. paths that are incapable of passing transitions. Identification of these paths helps in two ways. First, they save time for an Automatic Test Pattern Generator (ATPG) tool that tries to determine a test for this path, which is impossible, and secondly, they help in predicting the speed of the chip more accurately and help to do away with any pessimistic estimate that may reduce the yield of the chip. Testing of Path-delay faults, though very well-studied, is still quite a new field compared to stuck-at fault testing. Moreover it poses greater difficulty to the test engineers for the single fact that the number of paths in a circuit, may be exponential in the number of lines in the circuit, and hence, the difficulty in test generation can be easily envisaged. In this thesis, a very detailed theory is given how to model a path-delay fault in terms of an equivalent stuck-at fault(s) in a modified circuit derived from the original circuit. There exist several classification schemes for path-delay faults domain posed by various researchers. We describe a methodology for finding an equivalent stuck-at fault for each of these existing classifications [84, 87]. As the stuck-at fault domain is much better understood compared to the delay fault domain, this work will possibly provide a new direction in analyzing delay faults, and may offer an efficient tool for timing analysis in high performance circuits.

## 1.4 Organization of the thesis

*Chapter 2* presents the formulation and solutions of the monotone staircase cut (MSC) problem in a VLSI floorplan. Algorithms, complexity analyses, and experimental results are reported.

*Chapter 3* describes the proposed algorithm on topological routing and experimental results.

*Chapter 4* reports on our work on intersection minimization problem among several rectilinear Steiner trees.

*Chapter 5* presents the proposed geometric approach to thermal analysis of hot spots and reports the simulation results.

*Chapter 6* talks about the dilemma of using density or discrepancy in an ensemble of points for modeling hot spots. Relevant concepts and algorithms have been discussed in this chapter.

*Chapter 7* reports our findings on a new classification of path-delay faults in a circuit-under-test (CUT) in terms of stuck-at faults.

Finally, in *Chapter 8*, we summarize the work reported in this thesis and open discussions on some future problems to be settled.

# Chapter 2

## Floorplan-driven Partitioning

### 2.1 Introduction

A VLSI floorplan may be envisaged as a rectangular dissection of the bounding rectangle with isothetic cut lines. Each cell of the floorplan contains a circuit module. Two cut lines are assumed to meet at  $T$ -junctions only. A floorplan is said to be sliceable if it is obtained recursively by using through-cuts only, otherwise, it is non-sliceable. The non-sliceable floorplans usually pose more difficulty to the automated tools [17]. Floorplanning is a well-studied topic [129, 38] and is becoming very important in VLSI physical design as circuit size is growing very rapidly and hierarchical design with IP blocks is being widely used to reduce the design complexity [1]. Floorplan representation has also become an issue of fundamental importance in recent times [131, 18, 64]. There are two important characteristics of an efficient floorplan representation – the number of combinations of a representation and the time complexity of transformation between a representation and its corresponding floorplan. It is possible that two distinct combinations of a representation method actually correspond to the same floorplan. This is called the redundancy of representation, and a

method having less redundancy is more desirable [98, 108]. There are various combinatorial techniques, namely twin binary sequence, corner sequence, sequence pair, etc. to represent floorplan efficiently [75, 50, 120, 134, 133, 121]. Methods are available for the compact encoding of non-sliceable floorplans also [132, 19, 74]. Integrating the various steps like partitioning, floorplanning, and placement is one of the latest trends in physical design and is practiced in the analog domain also [2, 10, 68].

## 2.2 Floorplan partitioning problem

Consider a VLSI floorplan  $\mathcal{F}$  in Figure 2.1(a), containing a set of rectangular circuit blocks  $B = \{b_1, b_2, \dots, b_n\}$ , and a set of nets  $C = \{c_1, c_2, \dots, c_k\}$ . Each block is attached with few nets (shown in brackets) which are members in  $C$ . A *net* is a set of electrically equipotential terminals that are to be interconnected. Each block  $b_i$  has a netlist  $B_i (\subset C)$  attached to it. A staircase channel is an empty polygonal region on the silicon floor bounded by two monotonically increasing (or decreasing) staircase paths formed by connected isothetic line segments from one corner of the floor to its diagonally opposite corner (see Figure 2.1(a)). For simplicity, a staircase channel can be modeled by a staircase path from bottom-left corner of the floor to its top-right corner through the channel (see Figure 2.1(b)). This partitions the set of blocks in  $\mathcal{F}$  into two parts, namely *left-set* and *right-set* respectively. A *mincost staircase channel (MSC)* is a staircase channel such that the number of distinct nets attached to the blocks on both the *left-set* and *right-set*, is minimum.

We consider two distinct cases of this problem. The first one is the *two-terminal net problem*, where each net is attached to exactly two different blocks, and the objective is to locate a *MSC*. The problem is formulated using a variation of the max-flow-min-cut problem on a weighted digraph, and an

## Floorplan-driven Partitioning

---

algorithm of time complexity  $O(n \times k)$  is proposed, where  $n$  and  $k$  are the number of blocks and the nets respectively. Next, we consider the more general and realistic version of *MSC* problem where each net may have more than two terminals. The salient geometric features of this problem lead to a novel weighted digraph such that the min-cut of the graph gives the desired staircase that minimizes the number of *distinct* nets crossing it. The max-flow-min-cut algorithm run on this weighted digraph, leads to a solution of *multi-terminal MSC problem* in  $O((n+k) \times T)$  time, where  $T$  is the total number of terminals attached to all the blocks on the floor.

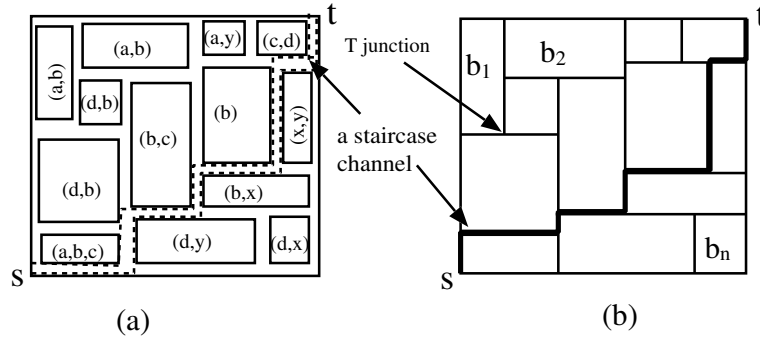


Figure 2.1: (a) A staircase channel in a layout and (b) A staircase in the corresponding floorplan

A relevant problem is to divide the floor into two partitions of almost equal area, and simultaneously minimizing the number of crossing nets [85]. This leads to a new mixed optimization problem, called *balanced mincost staircase partitioning*, where the objective function is a convex combination of the difference in the area of two partitions, and the number of crossing nets. But, staircase partitioning that minimizes the area difference is NP-hard even if minimization of the number of nets crossing the cut is not considered [85]. Hence, the mixed optimization problem stated above, is also computationally hard. An efficient heuristic for this problem is developed based on acyclic graph search with unrestricted edge cost [35]. Multiway partitioning with area

and pin constraints using max-flow min-cut paradigm has also been reported [76]. Another important optimization problem is to find a staircase channel which minimizes the difference in the number of blocks in the two partition. This can be solved in  $O(n)$  time [36].

### 2.3 Motivation for the *MSC* problem

The *MSC* problem is very pertinent to the repeater placement issues frequently confronted in interconnect-centric floorplanning used for deep-submicron VLSI physical design [29, 26, 102, 110, 106]. It is also relevant to the global routing problem [113].

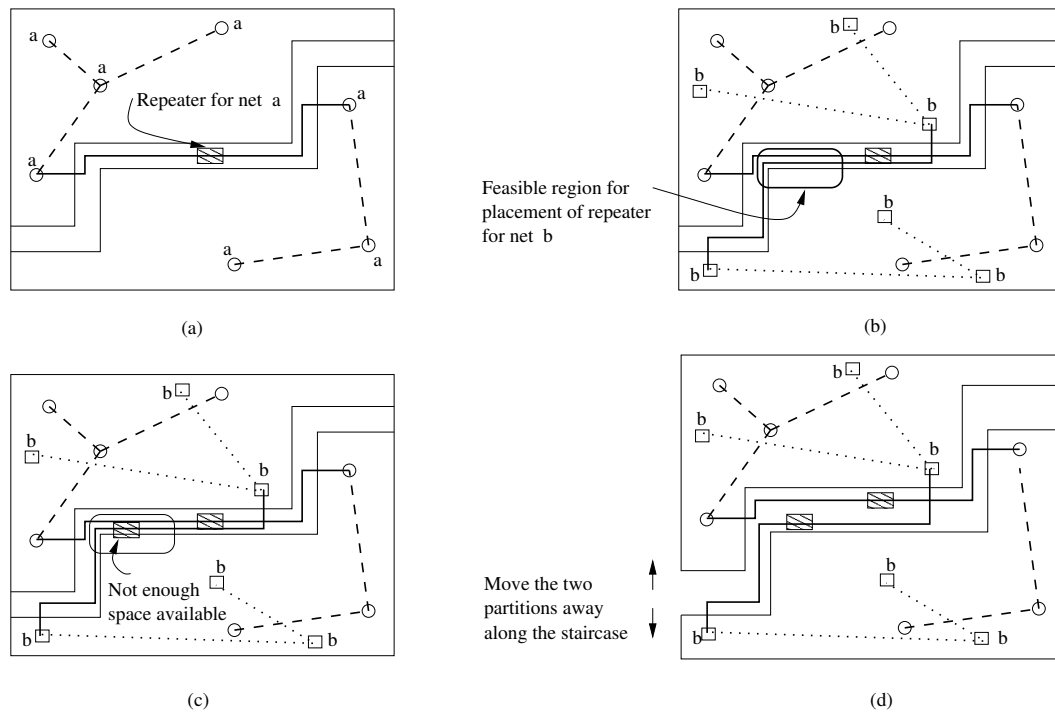


Figure 2.2: A typical example of repeater placement

### 2.3.1 Repeater placement problem

Due to the continued scaling of VLSI technologies, interconnects play a dominant role in determining system performance, power, reliability, and cost [27]. It has been observed that repeater block insertion is one of the most effective methods to optimize signal delay [29, 26, 102]. Judicious insertion of repeaters reduces the delay from quadratic to linear in terms of wire length [30]. Usually, most interconnect synthesis techniques are designed for post-placement interconnect optimization. Over 700K repeaters may be needed in a single chip for the 70 nm technology [28]. Insertion of many repeaters may significantly change the floorplan and placement of a design which is found to be optimum in the floorplanning stage. This leads to the problem of designing an efficient routability-driven repeater block placement method. Since repeaters for the signal nets are to be placed on the silicon layer, one needs to identify the unoccupied regions on the silicon layer. A recent method of repeater placement uses monotone paths [110] similar to monotone staircases. Minimizing the congestion in a monotone staircase channel, and hence the *MSC* problem, has the potential to strongly influence the repeater placement strategy.

In Figure 2.2 we show how the monotone staircase framework can help in repeater insertion through a typical example. The multi-terminal net  $a$  in Figure 2.2(a) has terminals on either side of a monotone staircase region, and it crosses the staircase only once. A repeater has to be inserted within the staircase region whenever the wire crossing the channel exceeds the permissible threshold. Note that the repeater is inserted in the device layer, whereas the wire passing through the repeater is in the metal layer. Further, for any net, there is a feasible region for repeater placement as indicated in Figure 2.2(b). It may be necessary to place more than one repeater, either for the same net or for different nets, within the same staircase region depending upon the length of the net(s). If the width(s) of a staircase region is not sufficient (Figure 2.2(c))

to accommodate all the repeaters required for meeting the timing requirements, the two partitions can be moved apart along the monotone staircase to create more space in the device layer, as shown in Figure 2.2(d). Monotone staircases offer this advantage as there does not exist any cyclic dependency among the channels in their routing order [45, 119]. Hence rip-up and re-route is never required during post-order traversal of the staircase tree, even if a particular channel needs to be stretched in order to accommodate more repeaters.

### 2.3.2 Channel/global routing

The hierarchy involved in the MSC problem is useful for solving both channel and global routing. A floorplan is sliceable if it is obtained recursively by using through-cuts only. Hierarchical partitioning by the cut lines may be used to facilitate global routing. Nonsliceability of floorplans lead to infeasible channel routing order [113, 119]. The problem of infeasible routing order in a general floorplan may be countered by either overestimating the width of certain channels, or completing the routing in multiple iterations. In the former case, the wasted area in the chip may increase hence a minimal set of channels needs to be chosen for widening, whereas the latter approach requires more CPU time.

If on the other hand the routing region is generalized to a monotone staircase, the routing order is always acyclic, and can be easily obtained by identifying staircase channels hierarchically [119]. A monotone channel also guarantees a feasible routing order and can be widened easily whereas for anon-monotone channel, stretching routing regions along the y-axis or x-axis by simply moving the two parts away may not always help. In addition, this may require rip-up-and-reroute. Thus the solution of the MSC problem offers a very good solution to channel routing. Efficient algorithms for routing through staircase

channels using manhattan-diagonal wiring model are available [34]. Routing may be done in a single iteration without overestimation of the width for any channels. A related problem of finding the widest staircase channel among a set of isothetic obstacles can be solved in  $O(n^2)$  time [100], where  $n$  is the number of vertices of all the obstacles placed on the floorplan. This can be applied to successful routing of maximum number of nets. Needless to say that with 5 or 6 metal layers in the modern VLSI technology, channel routing may be mainly relevant in the device layers but the size of the problem is significant.

The MSC problem is also relevant to the global routing problem. The staircase channels separating the terminals are identified in a hierarchical manner, and then the nets in the hierarchy are routed in a bottom-up fashion. The nets crossing the staircases which appear at the top of the hierarchy, are routed at the end, and hence should have less congestion in the corresponding portion of the metal layer. The nets in smaller partitions which cross the staircases towards the bottom of the hierarchy, need only local routing and are easier to route. Therefore, the hierarchy of the MSC problem identifies an order for global routing. The rationale behind choosing *minimal number of crossing nets* as one of the two optimization criteria is that minimization of routing congestion is highly desirable in global routing.

## 2.4 Graph-theoretic preliminaries

We define a directed graph (digraph)  $G = (V, E)$ , where the set of vertices  $V = \{v_1, v_2, \dots, v_n\}$  correspond to the set of blocks in  $B = \{b_1, b_2, \dots, b_n\}$  present on the floor  $\mathcal{F}$ . The two vertices  $v_1$  and  $v_n$  are designated as  $s$  (source) and  $t$  (sink). The blocks corresponding to  $s$  and  $t$  are  $b_1$  and  $b_n$  respectively, which are present at the top-left and bottom-right corners of the floor. Between two nodes  $v_i$  and  $v_j$  there is a directed edge  $e_{ij}$  if the corresponding blocks  $b_i$  and

$b_j$  are adjacent on the floor and block  $b_i$  appears either to the top or to the left of the block  $b_j$ . This graph will be referred to as *adjacency digraph* in the rest of this chapter.

**Lemma 2.1** *If  $e_{ij} \in E$  is a directed edge in  $G$ , then there exists at least one staircase channel in the floorplan such that the blocks  $b_i$  and  $b_j$  appear in the left-set and right-set respectively, and there exists no staircase with  $b_i$  in the right-set and  $b_j$  in the left-set.*

**Proof.** Follows from the construction of the adjacency digraph  $G$ . □

**Definition 2.1** *An  $st$ -cut is a cut in  $G$  that splits the set of nodes into two disjoint subsets, say  $S$  and  $T$  with  $s \in S$  and  $t \in T$ , such that for any node  $v \in S$ , there exists a path from  $s$  to  $v$  without passing through any node in  $T$ . Similarly, for any node  $v \in T$ , there exists a path from  $v$  to  $t$  without passing through any node in  $S$ .*

Thus, an  $st$ -cut corresponds to a partition of the set of blocks in  $B$  into exactly two disjoint subsets. The subset corresponding to  $S$  (resp.  $T$ ) contains block  $b_1$  (resp.  $b_n$ ). Henceforth, these two subsets (corresponding to  $S$  and  $T$ ) will be referred to as *left-set* and *right-set* respectively.

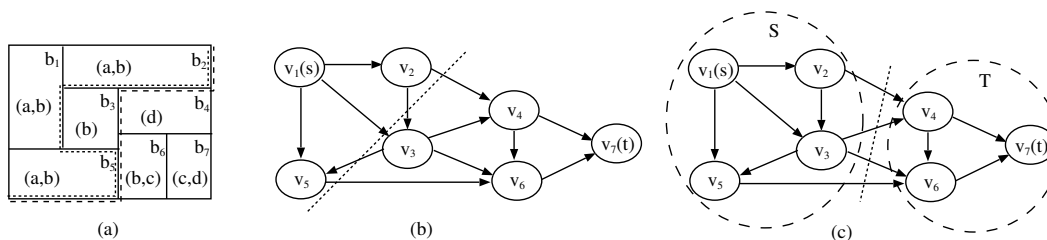


Figure 2.3: (a) A floorplan, (b) A non-monotone  $st$ -cut, and (c) A monotone  $st$ -cut of its adjacency digraph

In Figure 2.3(a), an example floor is shown, where the identification of each block appears at the top-right corner of the block. The nets present in

## Floorplan-driven Partitioning

---

each block are given within parentheses. Two examples of  $st$ -cut of its adjacency digraph are demonstrated in Figure 2.3(b) and 2.3(c). In Figure 2.3(a), the corresponding partitions of the floorplan are highlighted using dotted and dashed lines respectively. This demonstrates that an  $st$ -cut may not always partition the floor by monotone staircase from bottom-left corner of the floor to its top-right corner. Thus, in order to ensure that the resultant staircase to be monotone, we have to introduce the concept of monotone  $st$ -cut of  $G$ , as defined in the next sub-section.

### 2.4.1 Monotone $st$ -cut and staircase partitioning

**Definition 2.2** *An  $st$ -cut in  $G$  is said to be monotone if all the edges crossing the cut are directed from  $S$  to  $T$  (see Figure 2.3(c)).*

**Lemma 2.2** *Every node in  $G$  is reachable from  $s$ , and for any node  $v \in S$  obtained by a monotone  $st$ -cut, all the nodes present on the directed path from  $s$  to  $v$  also belongs to  $S$ .*

**Proof.** The graph  $G$  is connected and acyclic. The block  $b_1$  corresponding to the node  $s$  is either at the top or to the left of every other block. Now from the construction of  $G$  it easily follows that every node is reachable from  $s$ .

To prove the second part, let us assume for contradiction that a node  $v'$  belongs to the subset  $T$  and is present on the directed path from  $s$  to  $v$ . Now, there exists at least one edge in the subpath from  $v'$  to  $v$ , which is directed from a node in  $T$  to a node in  $S$ . This contradicts the fact that the cut is a *monotone  $st$ -cut*. □

Similarly, node  $t$  is reachable from every node in  $G$ , and for any node  $v \in T$ , all the nodes on the directed path from  $v$  to  $t$  will also belong to  $T$ .

**Theorem 2.1** *There exists a one-to-one correspondence between the set of staircase channels from the bottom-left corner to the top-right corner of the floor  $\mathcal{F}$ , and the set of monotone  $st$ -cuts in  $G$ .*

**Proof.** Let us consider a staircase channel. The set of vertices in  $G$  corresponding to the blocks in *left-set* and *right-set* are  $S$  and  $T$  respectively. Surely,  $S$  and  $T$  defines a cut in  $G$ , where  $s \in S$  and  $t \in T$ . Lemma 2.1, says that all the edges crossing over the above cut are directed from  $S$  to  $T$ . In other words, if  $e_{ij} = (v_i, v_j) \in E$  appears on the cut, then  $v_i \in S$  and  $v_j \in T$ . Thus, the cut is a monotone  $st$ -cut. The uniqueness of the cut follows from the fact that the set of nodes present in subset  $S$  uniquely determines the staircase.

Conversely, consider any *monotone  $st$ -cut* in  $G$ . By Lemma 2.2, the blocks corresponding to the nodes in  $S$  are separated from the blocks corresponding to the nodes in  $T$  by a single contiguous chain of alternately horizontal and vertical line segments from the *bottom-left* corner to the *top-right* corner. It remains to show that the chain is monotone.

Since the blocks are isothetic rectangles, the boundary of any cut changes its direction in  $T$ -junctions only, and can be expressed as a concatenation of a set of L-paths each member of which can be classified into any one of  $L_1$ ,  $L_2$ ,  $L_3$  and  $L_4$ , as shown in Figure 2.4.

Consider the two possible geometries of rectangular blocks around the L-path  $L_1$  (see Figure 2.4(a)). The blocks  $b_1$  and  $b_3$  are in the right side of  $L_1$  whereas  $b_2$  is in its left side. Clearly  $L_1$  can never be a part of a monotone  $st$ -cut due to the presence of an edge from  $T$  to  $S$  between the nodes corresponding to blocks  $b_1$  and  $b_2$ . Similarly  $L_2$  can not be observed in the chain generated by a monotone  $st$ -cut because of the edge between nodes corresponding to blocks  $b_2$  and  $b_3$  (see Figure 2.4(b)). However, Figures 2.4(c) and 2.4(d) respectively

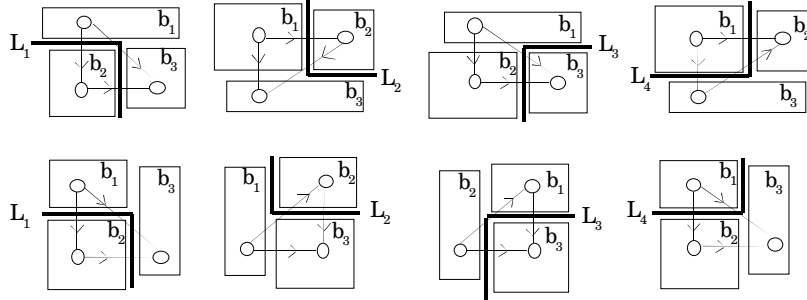


Figure 2.4: Four types of L-paths

show that all the edges between nodes corresponding to the blocks across  $L_3$  and  $L_4$  are directed from  $S$  to  $T$ , and hence they can be parts of a monotone  $st$ -cut. It can also be noticed that the chain containing only  $L_3$  and  $L_4$  is essentially monotone rising.  $\square$

**Corollary 2.1** *For any  $st$ -cut of  $G$ , if a directed edge exists from a node in  $T$  to a node in  $S$ , then the chain of isothetic line segments, partitioning the corresponding blocks, is non-monotone.*

## 2.5 Two-terminal net problem

In the *two-terminal net problem*, each net is connected with exactly two distinct blocks, and the objective is to recognize a staircase channel from the bottom-left corner of the floor to its top-right corner so that the number of nets appearing in both sides of the staircase is minimum. By Theorem 2.1, the problem reduces to finding a monotone  $st$ -cut of minimum cost in a suitably constructed graph; here the *cost* of a cut implies the maximum flow that can be sent from  $s$  to  $t$  through the edges on the cut. Efficient algorithm exists for finding the minimum cost  $st$ -cut of a weighted digraph [59]. But, the pro-

duced  $st$ -cut may not always be monotone ( see Figure 2.3(b)). In order to ensure that the minimum cost  $st$ -cut to be monotone, we need to construct a new graph  $G_{aug} = (V_{aug}, E_{aug})$  by augmenting the graph  $G$  as described in the following section.

### 2.5.1 Minimizing the number of crossing nets

The set of vertices in the augmented graph  $G_{aug}$  is same as that of the graph  $G$ , i.e.,  $V_{aug} = V$ . The set of edges and their weights are as follows: Let  $c_{ij}$  be the number of nets common between the pair of blocks  $b_i$  and  $b_j$ . Now consider the following cases if  $c_{ij}$  is non-zero.

**Case 1:**  $e_{ij} \in E$ . Here, the edge  $e_{ij}$  with capacity  $c_{ij}$  will be present in  $G_{aug}$  (see Figure 2.6(b)).

**Case 2:**  $e_{ij} \notin E$ , but a directed path from  $v_i$  to  $v_j$  is present in the graph  $G$ . This indicates that there does not exist any staircase which keeps  $b_i$  in the *right-set* and  $b_j$  in the *left-set* (see Figure 2.5(a)). So, we need to add a directed edge  $e_{ij}$  in  $E_{aug}$  with capacity  $c_{ij}$  (see Figure 2.6(c)).

**Case 3:** There does not exist any directed path from  $v_i$  to  $v_j$  or from  $v_j$  to  $v_i$  in  $G$ . This implies that any of the blocks  $b_i$  or  $b_j$  may appear on the *left-set* and the other in the *right-set* (see Figure 2.5(b)). So, we add a pair of directed edges  $e_{ij}$  and  $e_{ji}$  in  $E_{aug}$  with capacity  $c_{ij}$  (see Figure 2.6(d)) provided  $c_{ij} > 0$ .

Our intention is to find a minimum cost  $st$ -cut in the augmented graph  $G_{aug}$  such that the resulting cut is monotone. If we run the classical *single-source single-sink max-flow-min-cut* algorithm on the weighted digraph  $G_{aug}$ , it will produce a cut partitioning the set, such that all the forward edges crossing the cut will have their flow saturated, and all the back edges (if exist) will

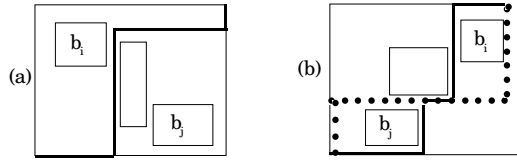


Figure 2.5: Illustration of weighted arcs in the graph  $G_{aug}$

have zero flow through them at the time of termination [44, 103, 123]. But the existence of such a back edge on the resulting cut causes the cut to be non-monotone, and by Corollary 2.1, the boundary of the partition separating the blocks corresponding to nodes in  $S$  and  $T$ , will be a non-monotone one. The arrival of back edges on the cut is prevented by executing the following augmentation step.

For each edge  $e_{ij} = (v_i, v_j) \in E$  in the adjacency digraph  $G$ , we add an edge  $e_{ji} = (v_j, v_i)$  with capacity  $\infty$  in the augmented graph  $G_{aug}$  (see Figure 2.6(e)).

Finally, we remove all the edges of capacity 0 from  $E_{aug}$  since no flow passes through them. In Figure 2.6, an example of a floorplan and the stepwise construction of its weighted digraph  $G_{aug}$  is demonstrated. Figure 2.6(f) shows the final  $G_{aug}$ .

**Definition 2.3** *An  $st$ -cut in  $G_{aug}$  is said to be monotone if the edges crossing over the cut having infinite capacity, are all directed from  $T$  to  $S$ .*

**Lemma 2.3** *There exists at least one monotone  $st$ -cut of finite cost in  $G_{aug}$ .*

**Proof.** Follows immediately by considering a trivial  $st$ -cut of  $G_{aug}$  with only the node  $s$ , corresponding to the block  $b_1$ , in the left partition  $S$  and all other nodes in the right partition  $T$ .  $\square$

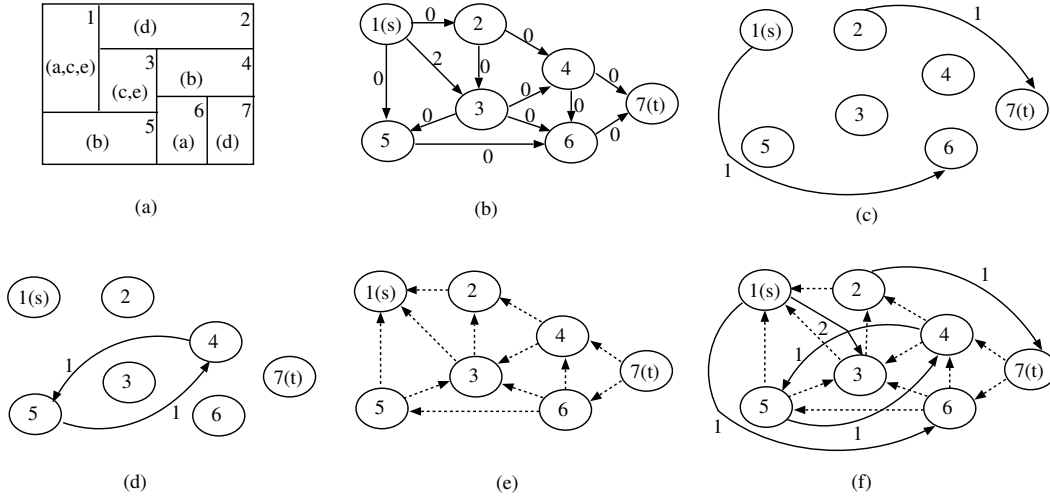


Figure 2.6: Stepwise construction of  $G_{aug}$  (capacity of dotted edges =  $\infty$ )

**Lemma 2.4** *For any non-monotone cut on the floorplan, the corresponding cut in  $G_{aug}$  has an infinite cost.*

**Proof.** The  $st$ -cut that produces a non-monotone path must have an edge  $e_{ij} \in E_{aug}$  directed from  $T$  to  $S$  (by Corollary 2.1). Hence the edge  $e_{ji}$ , which is directed from  $S$  to  $T$ , and having infinite cost will appear as a forward edge on the cut. If such a cut has to be the min-cut, the cost of the cut must have to be infinite, since on termination of the max-flow-min-cut algorithm all the edges in the forward direction has to be saturated.  $\square$

**Theorem 2.2** *The min-cut in  $G_{aug}$  is a monotone  $st$ -cut of minimum cost and it corresponds to the monotone staircase on the floorplan crossed by the minimum number of nets.*

**Proof.** Lemmas 2.3 and 2.4 says that the min-cut of  $G_{aug}$  is a non-monotone  $st$ -cut. Again Theorem 2.1 establishes a one-to-one correspondence between monotone  $st$ -cuts of  $G$  and the monotone staircases from bottom-left corner

## Floorplan-driven Partitioning

---

to the top-right corner of the floor. The same result is true for the graph  $G_{aug}$  also. Thus the min-cut in  $G_{aug}$  corresponds to the desired monotone staircase on the floor crossed by minimum number of nets.  $\square$

Theorem 2.2 leads to the following algorithm for finding the monotone staircase channel crossed by the minimum number of nets where each net has only two terminals attached to two different blocks on the floorplan.

### 2.5.2 Algorithm

Input: (i) list of blocks  $B$  with their geometric positions on the floor,  
(ii) list of nets  $C$ , and for each net  $c_\alpha$ , a pair of blocks where it appears.

Output: a staircase from the bottom-left corner of the floor to its top-right corner as a list of alternating horizontal and vertical line-segments.

Step 1: Construct the adjacency digraph  $G = (V, E)$  from the geometry of the floor.

Step 2: (\* Calculation of  $c_{ij}$ 's, the number of nets common to  $b_i$  and  $b_j$  \*)

Initially  $c_{ij}$  is set to 0 for all  $b_i, b_j \in B$ .

For each net  $c_\alpha \in C$ ,

If  $(b_i, b_j)$  be the pair of blocks containing net  $c_\alpha$ , add 1 to  $c_{ij}$ .

Step 3: (\* Construction of the augmented graph  $G_{aug} = (V_{aug}, E_{aug})$  \*)

Set  $V_{aug} = V$ .

if  $c_{ij} \neq 0$  then (\* Consider pair of blocks with  $c_{ij} \neq 0$  \*)

if  $e_{ij} \in E$  then put an edge  $e_{ij}$  with capacity  $c_{ij}$  in  $E_{aug}$ .

if  $e_{ij} \notin E$  then put 2 edges  $e_{ij}$  and  $e_{ji}$ , each with capacity  $c_{ij}$  in  $E_{aug}$ .

for each edge  $e_{ij} \in E$ , put an edge  $e_{ji}$  in  $E_{aug}$  with capacity  $\infty$ .

(\* The edges in  $G$  with  $c_{ij} = 0$  are not added in  $G_{aug}$  \*)

Step 4: Run the *Single source single sink max-flow-min-cut* algorithm [59] with  $s$  as source and  $t$  as sink. This will mark all the saturated edges.

Step 5: Find out the set of nodes  $S$  in the left partition that are reachable from  $s$  through unsaturated edges only, using *breadth first search*.

Step 6: (\* Generation of staircase \*)

For each node  $v_i \in S$ , check whether all its adjacent nodes in  $G$  also belong to  $S$ . If not, then it is a boundary node. Output the coordinates of the isothetic line segment(s) that separate(s) the corresponding block from the adjacent block(s) whose corresponding node(s) is(are) not in  $S$ .

### 2.5.3 Complexity analysis

Let the number of blocks on the floor be  $n$  and the number of nets be  $k$ . The construction of the adjacency digraph  $G$  in Step 1 and the computation of edge weights  $c_{ij}, \forall v_i, v_j \in V$  in Step 2 requires  $O(n)$  and  $O(k)$  time respectively since the graph is planar. The graph augmentation in Step 3 can be done in  $O(n+k)$  time. Step 4 can be performed in  $O(n \times k)$  time using the  $O(|V| \times |E|)$  time algorithm for the *single-source-single-sink* max-flow min-cut algorithm [59]. The *breadth-first search* in Step 5 and the generation of staircase cut in Step 6 requires  $O(k)$  and  $O(n)$  time respectively, in the worst case. Thus, the worst case time complexity of the above algorithm is  $O(n \times k)$ .

## 2.6 Multi-terminal net problem

In the global routing problem usually there exists nets having multiple terminals attached to more than two blocks. The model that we have proposed for the two-terminal nets, fails to work for the multi-terminal nets due to the following reason:

Consider a particular net  $c_\alpha$  which occurs in  $p$  distinct blocks. The job of the router is to connect the terminals of net  $c_\alpha$  placed in these  $p$  blocks. Now consider a monotone staircase that splits these  $p$  blocks such that the *left-set* contains  $m$  blocks and the *right-set* contains the remaining  $(p - m)$

blocks. The model proposed for *two-terminal nets* causes a contribution of a cost of  $m \times (p - m)$  on this staircase due to net  $c_\alpha$ . But usually a good global router connects all the terminals of  $c_\alpha$  inside a partition without crossing the staircase, and the inter-partition connection passes through the staircase only once. However, if there exists several clusters of terminals of the same net in both the partitions which are wide distance apart, it may have to cross the staircase more than once. But in that case also, the wire segments on the staircase of that net are routed using the same track. So, effectively we can say that each distinct net should contribute a unit cost to the staircase if it crosses the staircase, otherwise it should contribute zero. Thus our problem here is to locate the monotone staircase channels which will be crossed by the minimum number of distinct nets. In the following subsections, we discuss some geometric properties of the problem, and then we formulate the problem as a *network flow* problem on a suitably constructed graph.

### 2.6.1 Some geometric concepts

Let us consider the set of blocks containing the net  $c_\alpha$ . In Subsection 2.6.1, we show that some of these blocks need not be considered in connection to net  $c_\alpha$ , since the contribution of  $c_\alpha$  on each staircase remains same irrespective of the presence or absence of net  $c_\alpha$  in those blocks. By this way we can reduce the problem size to a large extent.

In Subsection 2.6.1, we shall discuss a geometric property of the problem that helps to construct a graph  $G_{aug}^M = (V_{aug}^M, E_{aug}^M)$  from the adjacency graph  $G$ , such that a monotone *st*-cut on  $G_{aug}^M$  should get unit cost for a particular net  $c_\alpha$  if and only if the blocks with net  $c_\alpha$  are present in both sides of the monotone staircase corresponding to that cut; otherwise, the contribution of net  $c_\alpha$  to the cost of the cut should be zero.

### Linkage elimination

Let  $b_i$ ,  $b_j$  and  $b_k$  be the three blocks on the floor where a particular net  $c_\alpha$  occurs, and in the adjacency digraph  $G$ , directed paths exist from  $v_i$  to  $v_j$  and from  $v_j$  to  $v_k$ . It can be easily observed that any monotone staircase that has  $c_\alpha$  on its both sides must have block  $b_i$  in the left-set, block  $b_k$  in the right set, and block  $b_j$  may reside in any side of the staircase. Thus we can easily ignore the net  $c_\alpha$  from the block  $b_j$ .

Let  $G^\alpha = (V^\alpha, E^\alpha)$  be the subgraph of  $G$  with the set of nodes such that the net  $c_\alpha$  appears in their corresponding blocks. Let  $V_{in}^\alpha$  and  $V_{out}^\alpha$  be the set of all indegree zero and outdegree zero nodes in  $G^\alpha$  respectively. The above discussion leads to the fact that one needs to consider only the set of nodes  $V_{in}^\alpha$  and  $V_{out}^\alpha$  or equivalently, the set of blocks  $B_{in}^\alpha$  and  $B_{out}^\alpha$ , respectively, in order to avoid the unnecessary blocks in connection with net  $c_\alpha$ .

### Cost estimation

Now we consider the net  $c_\alpha$  and the set of blocks  $B_{in}^\alpha$  and  $B_{out}^\alpha$  on the floorplan  $\mathcal{F}$ . It can be shown that there exists a unique staircase, which keeps all the blocks in  $B_{in}^\alpha$  to the right-set such that the area of the polygon bounded by this staircase, the *top* side and the *left* side of the floor, is maximum. We refer to this staircase as UPSTAIR( $\alpha$ ) and the polygon as  $P_{UP}$ . Similarly, a unique staircase exists keeping all the blocks in  $B_{out}^\alpha$  to the left-set such that the area of the polygon bounded by the staircase, the *right* side and the *bottom* side of the floor is maximum. This staircase is named as DOWNSTAIR( $\alpha$ ) and the corresponding polygon is named as  $P_{DOWN}$ . Let INTZONE( $\alpha$ ) be the area of the floor obtained by  $\mathcal{F} - P_{UP} - P_{DOWN}$ . The demonstration of UPSTAIR( $\alpha$ ), DOWNSTAIR( $\alpha$ ) and INTZONE( $\alpha$ ) for a net  $c_\alpha$  appears in Figure 2.7. In Figure 2.7(a), the region INTZONE( $\alpha$ ) is a simple staircase polygon, whereas

in Figure 2.7(b),  $\text{INTZONE}(\alpha)$  is not a simple staircase polygon.

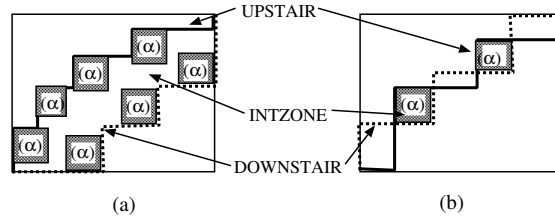


Figure 2.7: UPSTAIR, DOWNSTAIR and INTZONE for net  $c_\alpha$

**Observation 2.1** *A staircase, which is either entirely contained in the left-set of  $\text{UPSTAIR}(\alpha)$  or in the right-set of  $\text{DOWNSTAIR}(\alpha)$  will not have any cost due to  $c_\alpha$ . All other staircases on the floor which pass (partially or completely) through  $\text{INTZONE}(\alpha)$  will get a unit cost due to net  $c_\alpha$ .*

Let us now consider a pair of nets  $c_\alpha$  and  $c_\beta$  such that their INTZONES' overlap. If  $\text{INTZONE}(\alpha) \cap \text{INTZONE}(\beta)$  contains a single block, no staircase can pass through  $\text{INTZONE}(\alpha) \cap \text{INTZONE}(\beta)$ .

**Definition 2.4** *A pair of nets  $c_\alpha$  and  $c_\beta$  are said to be overlapping if the intersection region of  $\text{INTZONE}(\alpha)$  and  $\text{INTZONE}(\beta)$  contains at least two blocks (see Figure 2.8).*

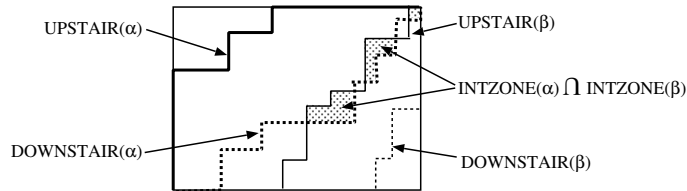


Figure 2.8: Overlap of INTZONES for two nets

If the INTZONE of a pair of nets  $c_\alpha$  and  $c_\beta$  are non-overlapping, there exists a staircase which can avoid passing through  $\text{INTZONE}(\alpha)$  and  $\text{INTZONE}(\beta)$ . In other words, for such a staircase, no cost is contributed by  $c_\alpha$  and  $c_\beta$ . Thus, the cost of the optimal staircase depends on the cardinality of the minimal subset of nets from set  $C$  such that their INTZONES' are pairwise overlapping.

## 2.6.2 Formulation

As in the two-terminal net problem, we now formulate the multi-terminal net problem as the location of a monotone *st*-cut of minimum cost in an weighted digraph appropriately constructed from the floor geometry. Consider the adjacency digraph  $G$  among the blocks of the floor. Below we describe the method of obtaining the augmented graph  $G_{aug}^M = (V_{aug}^M, E_{aug}^M)$  from graph  $G$  so that a monotone *st*-cut in  $G_{aug}^M$  should get unit cost for a particular net  $c_\alpha$  if and only if the blocks with net  $c_\alpha$  are present in both sides of the monotone staircase corresponding to that cut; otherwise the contribution of net  $c_\alpha$  to the cost of the cut should be zero.

### Graph augmentation

Let  $B^\alpha = \{b_1^\alpha, b_2^\alpha, \dots, b_r^\alpha\}$  be the set of blocks with which the net  $c_\alpha (\in C)$  is attached, and  $V^\alpha$  be the set of nodes in  $V_{aug}^M$  corresponding to the blocks in  $B^\alpha$ . We add two additional nodes  $s_\alpha$  and  $t_\alpha$  in  $V_{aug}^M$ , which will be referred to as *pseudo source* and *pseudo sink* for net  $c_\alpha$  respectively [81]. If however the block corresponding to the node  $s$  (resp.  $t$ ) contains the net  $c_\alpha$ ,  $s$  (resp.  $t$ ) itself will serve the purpose of  $s_\alpha$  (resp.  $t_\alpha$ ), and no new node need to be added. Thus the augmented set of nodes becomes  $V_{aug}^M = V \cup_{c_\alpha \in C} s_\alpha \cup_{c_\alpha \in C} t_\alpha$  with  $|V_{aug}^M| \leq n + 2k$ , where  $n$  and  $k$  are respectively the number of blocks and the number of distinct nets.

The stepwise construction of the graph  $G_{aug}^M$  corresponding to the floorplan of Figure 2.3 is shown in Figure 2.9. For convenience, the floorplan is shown again in Figure 2.9(a). The set of edges  $E_{aug}^M$  in the augmented graph is obtained as follows:

- Let  $\overline{E} = \{e_{ij} \mid e_{ij} \in E\}$ . For each element in  $\overline{E}$ , we put an edge in  $E_{aug}$  (see Figure 2.9(b)).

## Floorplan-driven Partitioning

---

- For each net  $c_\alpha$ , we add three sets of edges:
  - $E_s(\alpha)$  = set of edges from the nodes of  $V^\alpha$  to  $s_\alpha$  (see Figure 2.9(c)),
  - $E_t(\alpha)$  = set of edges from  $t_\alpha$  to the nodes in  $V^\alpha$  (see Figure 2.9(c)),
  - an edge  $e_{st}(\alpha)$  from  $s_\alpha$  to  $t_\alpha$  (see Figure 2.9(d)).

Thus,  $E_{aug}^M = \overline{E} \cup (\cup_{c_\alpha \in C} (E_s(\alpha) \cup E_t(\alpha) \cup \{e_{st}(\alpha)\}))$ . The final  $G_{aug}^M$  corresponding to the floorplan in Figure 2.9(a) is shown in Figure 2.9(e).

The number of edges in  $\overline{E}$  may be  $O(n)$ . The number of edges in  $\cup_{c_\alpha \in C} (E_s(\alpha) \cup E_t(\alpha))$  is  $O(T)$ , the total number of terminals considering all the blocks on the floor. The reason is that, for each block, the attached terminals correspond to different nets. The node corresponding to that block is added with  $s_\alpha$  and  $t_\alpha$ , for each net  $c_\alpha$  appearing in that block. The number of edges in  $\cup_{\alpha \in C} \{e_{st}(\alpha)\}$  is  $k$ , the number of nets present on the floor. Thus the total number of edges in  $E_{aug}^M$  is  $O(n + k + T) = O(T)$ .

For each  $\alpha$ , the *capacity* of the edge  $e_{st}(\alpha)$  is set to 1; the *capacity* of all other edges in  $E_{aug}^M$  are set to  $\infty$ .

This is important to mention that, we may further reduce the number of nodes of the graph following the method described in Subsection 2.6.1. In that case,  $V^\alpha$  will be equal to  $V_{in}^\alpha \cup V_{out}^\alpha$  for each net  $\alpha \in C$ .

### Properties of $G_{aug}^M$

Consider a unit flow through a path  $\Pi$  from source node  $s$  to the sink node  $t$  in  $G_{aug}^M$ . Note that,

The source  $s$  corresponds to the pseudo source node  $s_\alpha$  of some net  $c_\alpha$  that appears in block  $b_1$ .

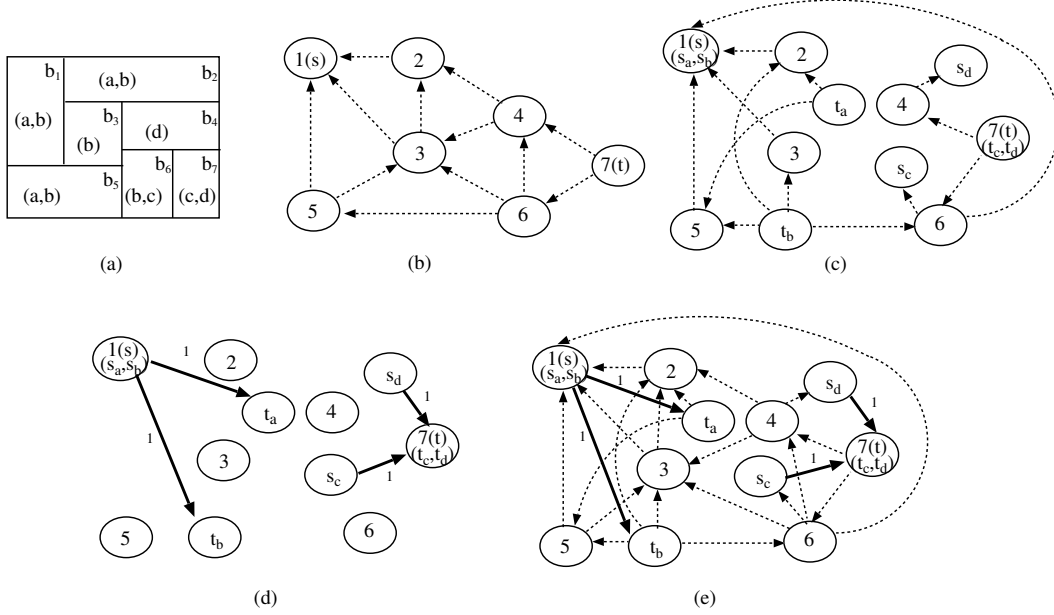


Figure 2.9: Stepwise construction of  $G_{aug}^M$  (capacity of dotted edges =  $\infty$ )

The flow will reach the pseudo sink node  $t_\alpha$  via edge  $(s_\alpha, t_\alpha) \in E_{aug}^M$ , and the capacity of the edge becomes saturated.

Again,  $t_\alpha$  is connected by back edges (of  $\infty$  capacity) to some nodes whose corresponding blocks contain net  $c_\alpha$ . So, the flow will reach to any one of these nodes such that another net  $c_\beta$  ( $\neq c_\alpha$ ) is attached to its corresponding block.

The flow will be passed from that node to  $s_\beta$  via a back edge of  $\infty$  capacity.

This process will be continued until the flow reaches the sink node  $t$ . Let  $(c_{\alpha_1}, c_{\alpha_2}, \dots, c_{\alpha_p})$  be the set of nets whose pseudo sources and pseudo sinks appear on path II. This implies that  $\text{INTZONE}(\alpha_1), \text{INTZONE}(\alpha_2), \dots, \text{INTZONE}(\alpha_p)$  are mutually pairwise overlapping as described in the Subsection 2.6.1. If at any stage, the flow gets stuck, this implies that there does not exist a minimal set of nets whose INTZONES' are pairwise overlapping.

**Definition 2.5** A monotone  $st$ -cut in  $G_{aug}^M$  is an  $st$ -cut that splits the set of nodes  $V_{aug}^M$  in two sets  $S^M$  and  $T^M$  such that  $s \in S^M$  and  $t \in T^M$ , and the

## Floorplan-driven Partitioning

---

edges crossing over the cut having infinite capacity are directed from  $T^M$  to  $S^M$ .

**Lemma 2.5** *Consider an arbitrary monotone  $st$ -cut in  $G_{aug}^M$ . If a node  $v_i$  whose corresponding block contains an instance of a net  $c_\alpha$  belongs to  $S^M$ , then the pseudo source  $s_\alpha$  will also belong to  $S^M$ . Similarly, for a node  $v_i$  whose corresponding block contains an instance of a net  $c_\alpha$  belongs to  $T^M$ , then the pseudo sink  $t_\alpha$  will also belong to  $T^M$ .*

**Proof.** On the contrary, let  $v_i \in S^M$  and  $s_\alpha \in T^M$ . Thus, the directed edge  $(v_i, s_\alpha)$  of infinite capacity will be directed from the *left-set* to the *right-set* of the floorplan which contradicts the fact that the cut is a monotone  $st$ -cut. Similarly if  $v_i$  belongs to the right partition  $T^M$ , then  $t_\alpha$  must be present in  $T^M$ .  $\square$

**Lemma 2.6** *In a monotone  $st$ -cut  $\chi_1$  in  $G_{aug}^M$ , if there exists a net  $c_\alpha$  such that its corresponding pseudo-source node  $s_\alpha$  and all the nodes in  $V^\alpha$  are in  $S^M$ , but the pseudo-sink node  $t_\alpha$  is in  $T^M$ , then there exists another monotone  $st$ -cut  $\chi_2$  with lesser cost where all the nodes in  $V^\alpha$  along with  $s_\alpha$  and  $t_\alpha$ , are in  $S^M$ .*

**Proof.** Let  $\chi_2$  be a cut obtained from  $\chi_1$  by replacing  $t_\alpha$  from  $T^M$  to  $S^M$ . We need to prove (i)  $\chi_2$  is a monotone  $st$ -cut, and (ii) the cost of  $\chi_2$  is lesser than that of  $\chi_1$ .

Part (i) is proved from the fact that  $\chi_1$  is a monotone  $st$ -cut, and all the edges from  $t_\alpha$  to the nodes in  $V^\alpha$  that appear as back edges in  $\chi_1$  are not appearing in  $\chi_2$ . Moreover, any other back edge that is not appearing on  $\chi_1$  will also not be present on  $\chi_2$ .

Part (ii) is proved from the fact that the forward edge from node  $s_\alpha$  to  $t_\alpha$ , which appears on  $\chi_1$  and contributes a unit cost, is absent in the cut  $\chi_2$ . Thus the cost of  $\chi_2$  is one less than that of  $\chi_1$ .  $\square$

Similarly it can be proved that for any monotone  $st$ -cut in  $G_{aug}^M$ , if for a particular net  $c_\alpha$ , the corresponding pseudo source node  $s_\alpha \in S^M$ , and all the nodes in  $V^\alpha$  along with the pseudo sink node  $t_\alpha \in T^M$ , then there exists another monotone  $st$ -cut with lesser cost where all the nodes in  $V^\alpha$  along with  $s_\alpha$  and  $t_\alpha$  are in  $T^M$ .

Lemma 6 leads to the concept of a restricted monotone  $st$ -cut as defined below.

**Definition 2.6** *A monotone  $st$ -cut is said to be restricted monotone  $st$ -cut if for a net  $c_\alpha$ , all the nodes of  $V^\alpha$  belongs to  $S^M$  (resp.  $T^M$ ), then both the nodes  $s_\alpha$  and  $t_\alpha$  will also belong to  $S^M$  (resp.  $T^M$ ).*

We should clearly mention that, for a particular net  $c_\alpha$ , both the partitions  $S^M$  and  $T^M$ , induced by a restricted monotone  $st$ -cut, may contain the nodes of  $V^\alpha$  (corresponding to  $c_\alpha$ ). In such a case,  $s_\alpha \in S^M$  and  $t_\alpha \in T^M$ . Lemma 2.7, stated below, indicates the cost of a restricted monotone  $st$ -cut in  $G_{aug}^M$ .

**Lemma 2.7** *Consider a restricted monotone  $st$ -cut of  $G_{aug}^M$ , say  $\chi$ , partitioning the set of nodes in  $V_{aug}^M$  into two sets, say  $S^M$  and  $T^M$ , such that for exactly  $p$  distinct nets, their pseudo source nodes appear in  $S^M$  and their pseudo sink nodes appear in  $T^M$ , and for all the remaining  $k - p$  nets, both the pseudo source and pseudo sink nodes of each of them appear in either of the sets  $S^M$  and  $T^M$ . Then the cost of  $\chi$  is exactly  $p$ .*

**Proof.** Follows from the construction of  $G_{aug}^M$ .  $\square$

## Floorplan-driven Partitioning

---

**Lemma 2.8** *For every monotone  $st$ -cut in the adjacency digraph  $G$ , there exists a unique restricted monotone  $st$ -cut in  $G_{aug}^M$ , and vice-versa.*

**Proof.** Consider any monotone  $st$ -cut in  $G$  which divides the set of nodes  $V$  into two disjoint partitions, say  $S$  and  $T$ . Now, consider a cut in  $G_{aug}^M$  such that its left partition  $S^M$  contains all the nodes of  $S$  and the right partition  $T^M$  contains all the nodes of  $T$ . Now we place the pseudo source nodes and pseudo sink nodes according to Lemma 2.5 and 2.6. From Lemma 2.5, it follows that the cut is a monotone  $st$ -cut. The uniqueness of the *restricted monotone  $st$ -cut* is guaranteed by Lemma 2.6.

The converse part of the Lemma is trivial since for any *restricted monotone  $st$ -cut* in  $G_{aug}^M$ , if we simply remove the pseudo sources and pseudo sinks from the two partitions  $S^M$  and  $T^M$ , the remaining nodes will determine the two partitions of nodes around a unique *monotone  $st$ -cut* of  $G$ .  $\square$

**Lemma 2.9** *For every monotone staircase in the floorplan, there exists a unique restricted monotone  $st$ -cut in  $G_{aug}^M$ , and the cost of the cut is exactly equal to the number of distinct nets crossed by that monotone staircase.*

**Proof.** The first part of the lemma follows from Theorem 2.1 and Lemma 2.8. The second part is a direct consequence of Lemma 2.7.  $\square$

**Theorem 2.3** *The min-cut in  $G_{aug}^M$  is a restricted monotone  $st$ -cut of minimum cost, and it corresponds to the monotone staircase on the floor crossed by the minimum number of distinct nets.*

**Proof.** Any non-monotone  $st$ -cut in  $G_{aug}^M$  will have an infinite cost due to the presence of an edge of infinite capacity as a forward edge. Such a cut will not be the output of a max-flow-min-cut algorithm. Again by Lemma 2.6, the min-cut will always correspond to a *restricted monotone  $st$ -cut*.  $\square$

### 2.6.3 Algorithm and its complexity analysis

Theorem 2.3 leads to the fact that in order to get a monotone staircase from bottom-left corner of the floor to its top-right corner so that the number of distinct nets appearing in both sides of the staircase is minimum, one has to find the minimum cost restricted monotone *st*-cut of the graph  $G_{aug}^M$  by running max-flow-min-cut algorithm on the graph  $G_{aug}^M$ . The algorithm consists of three steps:

**Step 1:** Construct the graph  $G_{aug}^M$  using the graph  $G$ , and the list of blocks attached with each net.

**Step 2:** Execute the max-flow-min-cut algorithm [59] on  $G_{aug}^M$ .

**Step 3:** Obtain the staircase from the set of blocks  $S^M$  obtained as left partition.

Step 1 of the algorithm requires  $O(n+k+T)$  time, which is the size of  $E_{aug}^M$ . In Step 2, the max-flow-min-cut algorithm [59] requires  $O((n+2k) \times (n+k+T))$  time which may be  $O(T \times (n+k))$  in the worst case, where the number of terminals is much larger than both the number of blocks and the number of distinct nets in the floor. As in the case of two-terminal net problem, Step 3 requires  $O(n)$  time. Thus, the overall time complexity of the algorithm is  $O(T \times (n+k))$ .

## 2.7 Hierarchical partitioning with staircases

A cross-country monotone staircase subdivides the rectangular floor into two subfloors. Now we consider each subfloor separately – some dummy blocks are considered to make the floor rectangular and the *mincost staircase* subdividing them is obtained. This process is repeated until each partition consists of a

## Floorplan-driven Partitioning

---

single block or no more valid staircases are present. Sometimes there may be more than one block in the partition that can be separated by straight channels only, and it happens more often down the hierarchy when the partitions are no longer rectangular. It can be easily shown that the total number of extra blocks needed in each level of hierarchy never exceeds twice the number of  $T$ -junctions that appear in the partitioning staircases up to the current level of hierarchy. In fact, while operating on graphs, addition of two dummy nodes (one acting as source for the right partition and the other as sink for the left partition) is enough for each level of hierarchy.

**Lemma 2.10** *The total number of  $T$ -junctions on the floor is  $2n - 2$ .*

**Proof.** Each  $T$ -junction consists of a corner of some rectangular block. We have a total of  $4n$  corners of  $n$  blocks; out of which 4 of them coincide with the corners of the floor. Thus we have a total of  $4n - 4$  corners and all of them are located in  $T$ -junctions. Again, a specific pair of corners designate a unique  $T$ -junction. So, the total number of  $T$ -junctions is  $2n - 2$ .  $\square$

**Theorem 2.4** *The number of disjoint monotone hierarchical staircases on a floor containing  $n$  blocks is equal to  $n - 1$ , irrespective of the way of generation of the hierarchy.*

**Proof.** At each  $T$ -junction, exactly one staircase ends and each staircase uses exactly two  $T$ -junctions at its two ends. Hence the number of disjoint staircases is exactly equal to  $\frac{2n-2}{2} = (n - 1)$ , whatever be the way of generation of the hierarchy.  $\square$

Note that at each level of the hierarchy, the blocks of all the subfloors constitute the entire floor. Let, at some level of hierarchy, the graph  $G = (V, E)$  be subdivided into  $p$  disjoint subgraphs  $G_i = (V_i, E_i)$ ,  $i = 1, 2, \dots, p$ , such

that  $V = \bigcup_{i=1}^p V_i$  and  $E = \bigcup_{i=1}^p E_i$ . By Lemma 2.10, the total number of extra blocks considered to make each floor rectangular, never exceeds  $O(n)$  in the worst case. So the total time requirement for running flow algorithms at that level will be of  $O(\sum_{i=1}^p |V_i| \times |E_i|) = O(|V| \times |E|)$ . If there are  $r$  levels in the hierarchy, the total time required will be of  $O(r \times |V| \times |E|)$ . Though the worst case complexity of  $r$  is  $O(n)$ , it may be  $O(\log n)$  if all the staircases found at each level of bipartition split the concerned subfloor into nearly equal number of blocks. In such a case, the overall time complexity of the two-terminal and multi-terminal version of the problem is  $O(nk \log n)$ , and  $O((n+k) \times T \times \log n)$  respectively. Next we give a methodology how the staircase partitions generated at the different levels of hierarchy can be made balanced so that the overall time complexity of this hierarchical scheme remains low.

## 2.8 Generation of a balanced hierarchy

A max-flow based algorithm [86] can determine in polynomial time a series of ms-cuts with minimum number of *crossing nets*, i.e., distinct nets with terminals on either side. These ms-cuts recursively bipartition a given rectangular floorplan until degenerate staircases of straight channels are obtained. Thus a hierarchy (binary tree) of ms-cuts is defined as shown in Figure 2.10, the channel number increasing with the depth of the hierarchy. The advantage in this scheme is that the problem of assigning nets to ms-cuts is solved by post-order traversal of this binary tree thereby reducing congestion in the longer ms-cuts towards the top of the tree and thus saves routing area. However, this algorithm more often yields an unbalanced tree, which reduces the efficacy of the scheme drastically. In this chapter, max-flow-based *balanced* bipartitioning method [59] is employed, as standard heuristics like Kernighan-Lin [113]

## Floorplan-driven Partitioning

---

cannot capture the topological constraints of the floorplan and thus guarantee ms-cut geometry.

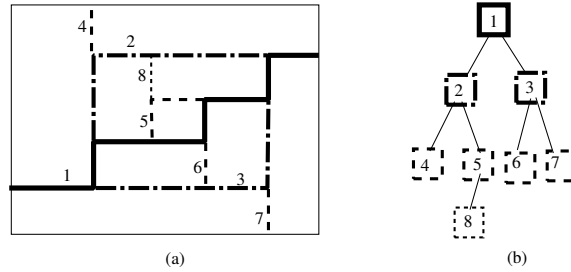


Figure 2.10: Hierarchy of monotone staircases in a floorplan

Two main criteria for balance in floorplan bipartitioning are (a) number of blocks, and (b) area of partitions. While the first one affects the depth of the hierarchy, the second one reflects that the routing area around a block depends on its area. The solutions for the two criteria differ considerably if the floorplan has few very large blocks and many small blocks. It is well-known [39] that even finding a number-balanced min-cut in a graph, disregarding the geometry of the cut, is NP-hard.

Next we establish that the problem of finding an area-balanced ms-cut in a floorplan is NP-complete even if minimization of cutsize is not considered. Then we propose an efficient max-flow based heuristic for generating a *balanced* hierarchy of ms-cuts, with as few crossing nets as possible. This method provides a framework in which either of the above two types of balanced hierarchy can be produced. As the problem involves optimizing the balance as well as the cutsize, our proposed method aims at optimizing a convex combination of the balance and the cutsize.

## 2.9 Problem formulation

In our routing model, given a floorplan  $F$  with rectangular boundary and rectangular blocks, the routing region is the set of lines defining the boundary of the blocks. Our approach is to define a hierarchy of ms-cuts comprising of these lines. In the detailed routing step [34], an ms-cut acquires finite width proportional to the number of tracks required for the nets assigned to it by the global router. With respect to staircase bipartitioning, a net is said to be *crossing* the staircase if it has terminals in both the partitions and the *cutsizes* of the staircase is equal to the number of crossing nets. Let  $n_l(n_r)$  and  $A_l(A_r)$  respectively denote the number and area of the blocks on the left (right) of the ms-cut.

*Fact* [81] : Given a floorplan with  $n$  blocks, (i) the total number of ms-cuts possible is exponential, (ii) a hierarchy of ms-cuts exists always with exactly  $n - 1$  ms-cuts.

Given  $F$  and its netlist, the following problems on ms-cuts may be formulated:

*Min-cut Problem  $P_c$*  : [86] an ms-cut with minimum cutsizes. (already dealt before in this chapter)

*Number-Balance Problem  $P_n$*  : [36] an ms-cut maximizing the number-balance ratio,  $\frac{\min(n_l, n_r)}{\max(n_l, n_r)}$ .

*Area-balance Problem  $P_A$*  : an ms-cut maximizing the area-balance ratio,  $\frac{\min(A_l, A_r)}{\max(A_l, A_r)}$ .

Noting that problems  $P_c$  and  $P_n$  are polynomial time solvable, the two problems studied here are the mixed optimization problems : *Number-balanced Min-cut  $P_{nc}$*  and *Area-balanced Min-cut  $P_{Ac}$*  . The goal is to find an ms-cut maximizing the cost(C), which is a convex combination of respective balance-ratio and cutsizes. We therefore define cost as  $C = \gamma \cdot (balance - ratio) + (1 -$

$\gamma) \cdot (1 - \text{cutsize}/\#\text{nets}), \gamma \in [0, 1]$ .

The parameter  $\gamma$  is termed as *control factor* which determines the trade-off between balance-ratio of the bi-partition and the cut-size. For  $\gamma = 0$ , the cost  $C$  is nothing but the min-cut. Both  $P_{nc}$  and  $P_{Ac}$  reduce to  $P_c$ . Similarly, for  $\gamma = 1$ , the emphasis is on balance alone so the two problems reduce to  $P_n$  or  $P_A$  depending on the type of balance-ratio.

## 2.10 Area-balanced monotone staircase bipartitioning

We prove that problem  $P_{Ac}$  is NP-hard by showing that problem  $P_A$ , the special case with  $\gamma = 1$ , is NP-hard.

Let us consider the decision problem  $P'_A$  corresponding to  $P_A$ , namely, *does there exist an ms-cut in a given floorplan  $F$  such that  $A_l = A_r$ ?* Let us first show that  $P'_A \in \text{NP}$ . We assume that all junctions in the floorplan are  $T$ -junctions, a cross-junction being a pair of  $T$ -junctions with an infinitesimal skew. Starting from  $P$ , the *bottom-left* corner of  $F$ , we move either upward or to the right. At each  $T$ -junction encountered, we choose non-deterministically either  $+x$  or  $+y$  direction if both options exist, till we reach  $Q$ , the *top-right* corner. The path traversed is a *rising ms-cut*. The time taken is  $O(n)$  as the number of operations is equal to the number of  $T$ -junctions visited, and the total number of  $T$ -junctions in  $F$  is  $2n - 2$  [81]. Hence  $P'_A \in \text{NP}$ . The NP-completeness of  $P'_A$  follows from the polynomial reduction of the Set Partitioning problem [39] to  $P'_A$ .

**Definition 2.7** *Set Partitioning Problem (SPP) : Given a finite set  $A$  and a size  $s(a) \in \mathbb{Z}^+$  for each  $a \in A$ . Does there exist a subset  $A' \subseteq A$  such that  $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$ ?*

Given a set  $A = \{a_1, a_2, \dots, a_k\}$ , we construct an equivalent instance of the problem  $P'_A$  in two stages. First, we place a set  $S = \{b_1, b_2, \dots, b_k\}$  of square blocks along the diagonal of a rectangular floorplan (Figure 2.11(a)), where area of  $b_i$  is equal to  $s(a_i)$  for each  $a_i \in A$ . In order to cover the entire floorplan we place a set  $B_{adj} = \bigcup_{i=1}^{k-1} \{b_{iL}, b_{iR}\}$  of additional blocks on the floorplan. As  $b_i$ 's are of square-shaped, we clearly have  $\text{Area}(b_{iL}) = \text{Area}(b_{iR})$  for  $1 \leq i \leq k-1$ .

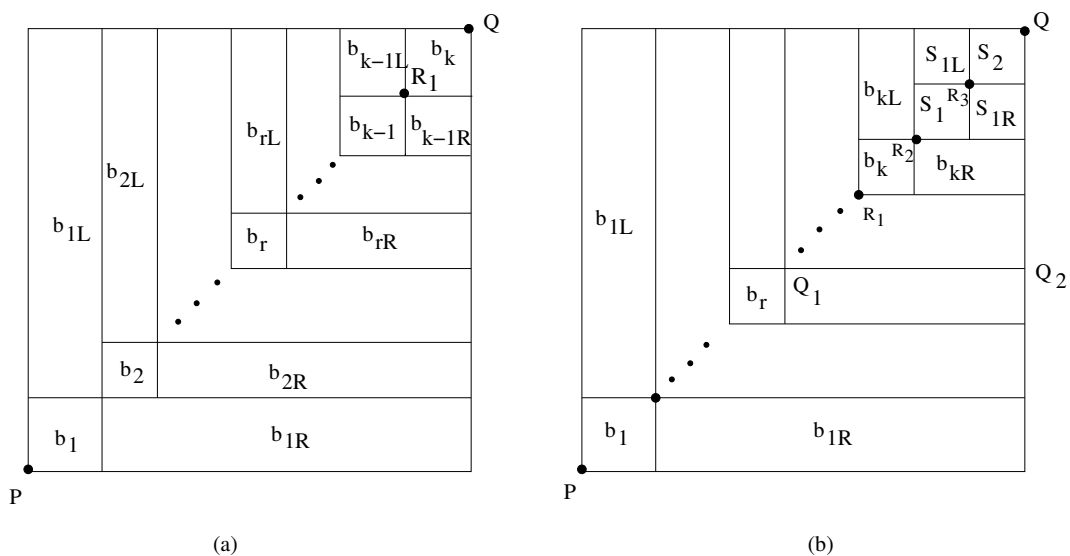


Figure 2.11: Floorplans for proof of NP-completeness

Now for any rising ms-cut, there is a corresponding partitioning of the set  $A$ . Our intention is to find the *Area-balanced* ms-cut for which the difference in the weight of the two corresponding partitions of  $A$  is minimum. This correspondence holds if and only if the ms-cut passes through the point  $R_1$  (Figure 2.11(a)), because in that case for all  $i \in [1, k]$ ,  $b_{iL}$  and  $b_{iR}$  respectively belongs to the left and right partitions and hence the total area of additional blocks in either partition is equal, which ensures that the additional blocks cannot bias any partition.

In the second stage (Figure 2.11(b)), we ensure that the *Area-balanced* ms-

## Floorplan-driven Partitioning

---

cut passes through  $R_1$  by introducing two more square blocks  $S_1$  and  $S_2$  along the diagonal such that

$$K' = \text{Area}(S_1) = \text{Area}(S_2) > \sum_{a_i \in A} s(a_i) = K.$$

Finally, to obtain the desired floorplan  $F$ , we add four more additional blocks  $b_{kL}$ ,  $b_{kR}$ ,  $S_{1L}$  and  $S_{1R}$ .

**Lemma 2.11** *An Area-balanced ms-cut  $\chi$  in  $F$  cannot have both  $S_1$  and  $S_2$  in the same partition and must pass through the points  $R_1$ ,  $R_2$  and  $R_3$  (Figure 2.11(b)).*

**Proof.** Suppose  $\chi$  passes through the points  $Q_1$  and  $Q_2$  keeping both  $S_1$  and  $S_2$  in its left partition. Even if all the blocks  $b_1, b_2, \dots, b_r$  lie in the right partition, the difference in area between the two partitions  $|A_l - A_r| \geq 2K' - K > K'$ . But for any ms-cut that keeps blocks  $S_1$  and  $S_2$  in different partitions, we have  $|A_l - A_r| \leq K < K'$ .

Hence  $S_1$  and  $S_2$  must be on different sides of  $\chi$  to achieve area-balance; the rest follows easily.  $\square$

The above lemma shows that for an *Area-balanced* ms-cut the additional blocks do not bias any partition.

The reduction is completed by replacing each of the cross-junctions in the floorplan by two  $T$ -junctions separated by a distance  $d$ . The difference in areas of the two partitions may increase as for any  $i$ ,  $b_{iL} = b_{iR}$  is no longer true. But if we keep  $d < \delta$ , a finite upper bound, the correspondence between the solutions to SPP and  $P'_A$  still remains. In *SPP* if we denote the difference in the sum of integers of the two subsets as the cost of a solution then two solutions with unequal cost should differ in their costs by at least 2, e.g. if for an instance of SPP, the optimal solution has a cost of 0, i.e. there exists two subsets with equal sum, then there cannot exist a solution of cost 1. On the

other hand it can be shown that the difference in areas of the two partitions can be made  $\leq 1$  by selecting  $\delta$  to be a polynomial-time computable function of all  $s(a_i)$ ,  $a_i \in A$ , and hence an *area-balanced* ms-cut in  $F$  corresponds exactly to a solution of SPP for set  $A$ . The time required to construct such an instance of the floorplan is  $O(|A|)$ . The computation of  $\delta$  is explained below and is also shown in Figure 2.12.

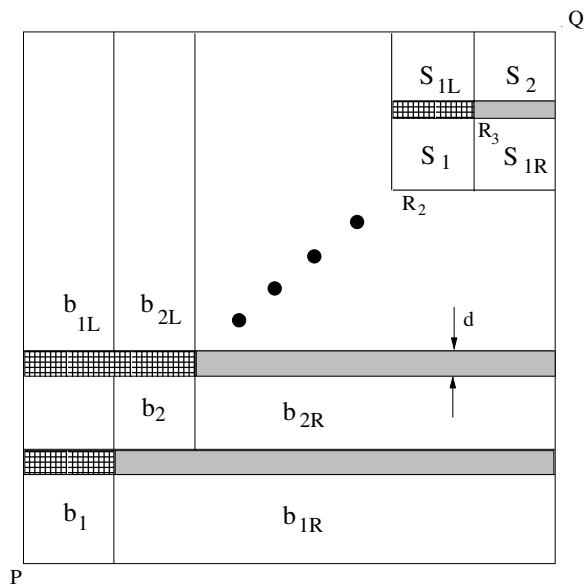


Figure 2.12: Computation of  $\delta$

**Upper Bound on  $d$ :**

$$A_l - A_r = \{ d((k+1)c_1 + kc_2 + \dots + 2c_k + C) \} - \{ d(c_2 + 2c_3 + \dots + (k-1)c_k + kC + (k+1)C) \},$$

where  $c_i = \sqrt{s(a_i)}$ ,  $1 \leq i \leq k$  and  $C = \sqrt{K'}$ .

So to keep  $|A_l - A_r| < 1$ , we define  $\delta$  to be less than  $1/(((k+1)c_1 + k.c_2 + \dots + 2c_k + C) - (c_2 + 2c_3 + \dots + (k-1)c_k + k.C + (k+1)C))$ .

Finally we arrive at the following theorem.

**Theorem 2.5** *Problem  $P'_A$  is NP-complete.*

**Proof.** Consider an *area-balanced monotone staircase* in the floorplan con-

structured above. This corresponds to a partitioning of  $A$  for which the difference in the sum of integers in the two partitions will be minimum and vice versa. The time required to construct such an instance of the floorplan is  $O(|A|)$ . Hence  $P'_A$  is NP-hard and as we already have  $P'_A \in \text{NP}$ ,  $P'_A$  is NP-complete.  $\square$

## 2.11 Heuristic for mixed optimization

The NP-hardness of problem  $P_{Ac}$  mandates the design of an efficient heuristic for solving it. At this juncture, we also state the following:

**Conjecture 2.1** *Problem  $P_{nc}$  is NP-hard.*

For the problems  $P_{nc}$  and  $P_{Ac}$ , a common heuristic framework is therefore proposed. An instance of  $P_{Ac}$  with unit area blocks is equivalent to an instance of  $P_{nc}$ . As we require a balanced ms-cut, we construct an augmented graph  $G_{aug}$  exactly in the same fashion as we have done earlier for problem  $P_c$  [86], to ensure ms-cut and apply a max-flow based balanced bipartitioning method [130]. For brevity, even for a floorplan with multi-terminal nets we refer to this graph as  $G_{aug}$  only instead of  $G_{aug}^M$ .

### 2.11.1 Flow based balanced bipartitioning

Given a network having weights associated with its nodes, a bipartition is said to be balanced if the weights of the two parts are each equal to  $W/2$ , where  $W$  is the total weight associated with all its nodes. The key idea is to improve the balance ratio of a bipartition incrementally by determining the min-cut [59] on the given network, then collapsing all nodes in the smaller partition with a randomly chosen node of the other part, and finding the min-cut for the collapsed network. This can be implemented efficiently by computing the

flow in the collapsed network incrementally instead of starting the max-flow algorithm all over again. The algorithm terminates when the weight of the left partition lies between  $(1 - \epsilon)W/2$  and  $(1 + \epsilon)W/2$  where the deviation factor  $\epsilon < 1$  is chosen to be very small. It was proved [130] that the number of iterations and the final cutsize are non-increasing functions of  $\epsilon$ .

### 2.11.2 Overview of our heuristic

The inputs to the heuristic for problems  $P_{nc}$  and  $P_{Ac}$  are (i) a floorplan  $F$  with its netlist  $N$ , (ii) parameter  $\gamma$ , (iii) desired type of balance *btype*, and (iv)  $\epsilon$ . The output is an ms-cut that satisfies the balance criterion as well as optimizes the convex cost  $C$ , as defined earlier.

*Algorithm MOP(F)*

- Step 1.** Construct  $G_{aug}$  for given  $F$ .
- Step 2.** Find a min-cut  $\chi'$  using a max-flow algorithm on  $G_{aug}$ .
- Step 3.** Compute balance ratio depending on *btype*.
- Step 4.** If the cut  $\chi'$  is balanced, then go to Step 7.
- Step 5.** Store  $\chi'$  and its cost; collapse all nodes in the smaller part of  $G_{aug}$  corresponding to  $\chi'$  to a node randomly chosen from the other part, adjacent to  $\chi'$ , to form the new source or sink accordingly.
- Step 6.** Go to Step 2.
- Step 7.** From the list of stored cuts, return the one with maximum cost.

The cut thus found yields two subgraphs of  $G_{aug}$  for which with minor adjustments for definition of source and sink, the above procedure can be repeated. Finally, a balanced hierarchy of ms-cuts is obtained by recursively

## Floorplan-driven Partitioning

---

calling the algorithm *MOP* on each of the two subgraphs obtained, till floorplans corresponding to the subgraphs have no non-degenerate staircases. The main algorithm *Staircase\_Hierarchy* is given below.

*Algorithm Staircase\_hierarchy*

*Input:* A graph  $G_{aug}$  corresponding to given floorplan  $F$ .

*Output:* A tree of monotone staircases  $S_G$ .

*begin*

**Step 1.** If  $F$  has more than 2 blocks and at least one valid staircase exists, then  $c := MOP(F)$  else return NULL.

**Step 2.** Determine  $s_G$ , the ms-cut in  $F$  corresponding to  $c$ , producing the two sub-floorplans  $F_L$  and  $F_R$ .

**Step 3.**  $S_L := \text{Staircase\_hierarchy}(F_L)$

**Step 4.**  $S_R := \text{Staircase\_hierarchy}(F_R)$

**Step 5.** Return a staircase tree  $S_G$  with  $s_G$  as its root,  $S_L$  and  $S_R$  respectively the left and right subtree of  $s_G$ .

*end*

## 2.12 Experimental results

We have implemented the heuristic for the mixed optimization problem and have run it on the five MCNC floorplan benchmarks given in Table 2.1. We have been able to generate the hierarchy of ms-cuts in each case. Further, the effect of the control parameter  $\gamma$  on the trade-off between number of nets

Table 2.1: Details of MCNC floorplan benchmarks

Name	# blocks	# nets
apte	9	97
xerox	10	208
hp	11	83
ami33	33	123
ami49	49	408

crossing *ms-cut* and the *balance-ratio* is evident from the results presented below.

As a typical case, we present the results of the entire run of our proposed heuristic for floorplan *ami33* in Tables 2.2 and 2.3 for *number-balance* and *area-balance* respectively. For brevity, we have omitted the detailed tables for the other benchmark floorplans but the outcome of our experiments are presented in a comprehensive manner in the plots shown in Figure 2.13. Finally, Table 2.4 gives an idea about the time taken by our heuristic.

We have tested for 6 different values of  $\gamma$  on each of the benchmarks. One of the important goals of these experiments is to determine a suitable value of  $\gamma$ , which can then be used for other floorplans. In Table 2,  $\gamma$  is varied from 0 to 1 in increments of 0.2. We repeat here that  $\gamma = 0$  corresponds to minimum *cutsizes* without *balance*, and  $\gamma = 1$  corresponds to *balanced bipartition*. The values of  $\gamma$  in between 0 and 1 leads to the *mixed optimization problem*. The 2nd column *ms-cut #* gives the *depth of the hierarchy of the cuts generated*. When  $\gamma = 0$ , the cuts generated are highly *unbalanced* and at each level we have only one *staircase*. But for  $\gamma > 0$ , each level in the hierarchy except the first has more than one *ms-cuts*, eg. level 2 has two *ms-cuts* 2a and 2b. The next two columns give the *number of blocks and nets being partitioned by the ms-cut at that node of the hierarchy*. The column *Balance ratio* gives the

## Floorplan-driven Partitioning

Table 2.2: Effect of  $\gamma$  on number-balanced hierarchy for *ami33*

$\gamma$	ms-cut #	# of blks	# of nets	Balance ratio	Net cut	Cost	$\gamma$	ms-cut #	# of blks	# of nets	Balance ratio	Net cut	Cost	
0	1	33	80	1/32	5	0.938	0.2	1	33	80	1/32	5	0.756	
	2	32	79	1/31	2	0.975		2	32	79	1/31	2	0.786	
	3	31	79	1/30	7	0.911		3	31	79	1/30	7	0.736	
	4	30	76	1/29	5	0.934		4	30	76	1/29	5	0.754	
	5	29	74	1/28	8	0.892		5	29	74	1/28	8	0.721	
	6	28	72	1/27	5	0.931		6	28	72	1/27	5	0.752	
	7	27	71	1/26	12	0.831		7	27	71	6/21	13	0.711	
	8	26	65	1/25	7	0.892		8a	6	12	3/3	7	0.533	
	9	25	64	1/24	4	0.938		8b	21	58	1/20	7	0.713	
	10	24	64	1/23	5	0.922		9	20	55	1/19	6	0.723	
	11	23	61	1/22	7	0.885		10	19	54	1/18	7	0.707	
	12	22	60	1/21	7	0.883		11	18	51	2/16	7	0.715	
	13	21	57	1/20	8	0.860		12	16	47	1/15	8	0.677	
	14	20	55	1/19	7	0.873		13	15	44	1/14	6	0.705	
	15	19	52	1/18	6	0.885		14	14	40	3/11	7	0.715	
	16	18	51	1/17	8	0.843		15	11	34	3/8	6	0.734	
	17	17	48	2/15	7	0.854		16	8	32	1/7	5	0.704	
	18	15	44	2/13	6	0.864		17	7	31	3/4	6	0.795	
	19	13	38	1/12	6	0.842		18	4	25	1/3	16	0.355	
	20	12	34	1/11	3	0.912								
	21	11	34	2/9	5	0.853								
	22	9	33	1/8	5	0.848								
	23	8	32	1/7	5	0.844								
	24	7	31	1/6	16	0.484								
	25	6	15	2/4	3	0.800								
	26	4	10	2/2	7	0.300								
0.4	1	33	80	9/24	21	0.593	0.6	1	33	80	13/20	37	0.605	
	2a	24	61	11/13	21	0.732		2a	13	18	5/8	12	0.508	
	3a	11	16	2/9	4	0.539		2b	20	46	8/12	18	0.643	
	3b	13	35	3/10	6	0.617		3a	8	12	4/4	9	0.700	
	4a	9	16	4/5	8	0.620		3b	12	24	5/7	8	0.695	
	4b	10	33	3/7	9	0.608		4a	4	4	2/2	4	0.600	
	5	4	8	2/2	8	0.400		4b	5	20	2/3	18	0.440	
0.8	1	33	80	13/20	37	0.628	1	1	33	80	13/20	37	0.650	
	2a	13	18	5/8	12	0.567		2a	13	18	5/8	12	0.625	
	2b	20	46	8/12	18	0.655		2b	20	46	8/12	18	0.667	
	3a	8	12	4/4	9	0.850		3a	8	12	4/4	9	1.000	
	3b	12	24	5/7	8	0.705		3b	12	24	5/7	8	0.714	
	4a	4	4	2/2	4	0.800		4a	4	4	2/2	4	1.000	
	4b	5	20	2/3	18	0.553		4b	5	20	2/3	18	0.667	

relative distribution of the number of blocks on either side of the cut that is being generated. The column *Net cut* gives the number of nets crossing the cut and the final column gives the cost of mixed optimization as defined in Section 3.

Table 3 is very similar to Table 2 with only one additional column *Area*

## Floorplan-driven Partitioning

Table 2.3: Effect of  $\gamma$  on area-balanced hierarchy for *ami33*

$\gamma$	cut	#	#	Num	Area	Net	Cost	$\gamma$	cut	#	#	Num	Area	Net	Cost
	#	blks	nets	ratio	ratio	cut			#	blks	nets	ratio	ratio	cut	
0	1	33	80	1/32	0.022	5	0.938	0.2	1	33	80	1/32	0.022	5	0.754
	2	32	79	1/31	0.011	2	0.975		2	32	79	1/31	0.011	2	0.782
	3	31	79	1/30	0.005	7	0.911		3	31	79	1/30	0.005	7	0.730
	4	30	76	1/29	0.077	5	0.934		4	30	76	1/29	0.077	5	0.763
	5	29	74	1/28	0.035	8	0.892		5	29	74	1/28	0.035	8	0.721
	6	28	72	1/27	0.015	5	0.931		6	28	72	1/27	0.015	5	0.747
	7	27	71	1/26	0.040	12	0.831		7	27	71	6/21	0.304	13	0.714
	8	26	65	1/25	0.019	7	0.892		8a	6	12	3/3	0.447	7	0.423
	9	25	64	1/24	0.046	4	0.938		8b	21	58	1/20	0.079	7	0.719
	10	24	64	1/23	0.013	5	0.922		9	20	55	1/19	0.068	6	0.726
	11	23	61	1/22	0.027	7	0.885		10	19	54	1/18	0.023	7	0.701
	12	22	60	1/21	0.069	7	0.883		11	18	51	2/16	0.038	7	0.698
	13	21	57	1/20	0.137	8	0.860		12	16	47	1/15	0.033	8	0.670
	14	20	55	1/19	0.021	7	0.873		13	15	44	1/14	0.013	6	0.694
	15	19	52	1/18	0.069	6	0.885		14	14	40	2/12	0.143	6	0.709
	16	18	51	1/17	0.031	8	0.843		15	12	34	3/9	0.865	6	0.832
	17	17	48	2/15	0.039	7	0.854		16	9	32	1/8	0.166	3	0.758
	18	15	44	2/13	0.140	6	0.864		17	8	32	4/4	0.662	7	0.757
	19	13	38	1/12	0.015	6	0.842		18	4	25	1/3	0.593	16	0.407
	20	12	34	1/11	0.082	3	0.912								
	21	11	34	2/9	0.640	5	0.853								
	22	9	33	1/8	0.113	5	0.848								
	23	8	32	1/7	0.255	5	0.844								
	24	7	31	1/6	0.343	16	0.484								
	25	6	15	2/4	0.804	3	0.800								
	26	4	10	2/2	0.021	7	0.300								
0.4	1	33	80	13/20	0.696	37	0.601	0.6	1	33	80	13/20	0.696	37	0.632
	2a	13	18	1/12	0.202	3	0.581		2a	13	18	6/7	0.792	13	0.586
	2b	20	46	8/12	0.682	22	0.586		2b	20	46	8/12	0.682	22	0.618
	3a	12	16	4/8	0.966	10	0.611		3a	7	6	3/4	0.919	2	0.818
	3b	8	19	3/5	0.695	4	0.752		3b	6	6	2/4	0.555	2	0.600
	4a	4	7	1/3	0.990	7	0.396		3c	8	19	3/5	0.695	4	0.733
	4b	8	4	3/5	0.943	3	0.527		4	5	19	2/3	0.798	18	0.500
	4c	5	19	1/4	0.482	14	0.351								
0.8	1	33	80	13/20	0.696	37	0.664	1	1	33	80	13/20	0.696	37	0.696
	2a	13	18	6/7	0.792	13	0.689		2a	13	18	6/7	0.792	13	0.792
	2b	20	46	8/12	0.682	22	0.650		2b	20	46	8/12	0.682	22	0.682
	3a	7	6	3/4	0.919	2	0.868		3a	7	6	3/4	0.919	2	0.919
	3b	6	6	3/3	0.703	4	0.629		3b	6	6	3/3	0.703	4	0.703
	3c	8	19	3/5	0.695	4	0.714		3c	8	19	3/5	0.695	4	0.695
	4	5	19	2/3	0.798	18	0.649		4	5	19	2/3	0.798	18	0.798

*ratio*, where the balance of areas of the two partitions generated by each ms-cut is also shown. In both these tables the following facts are observed quite naturally. With increase in  $\gamma$ , balance factor increases, which in turn brings down the number of levels of hierarchy, whereas the number of nets crossing the cut increases. For interpreting the effect of trade-off control parameter  $\gamma$

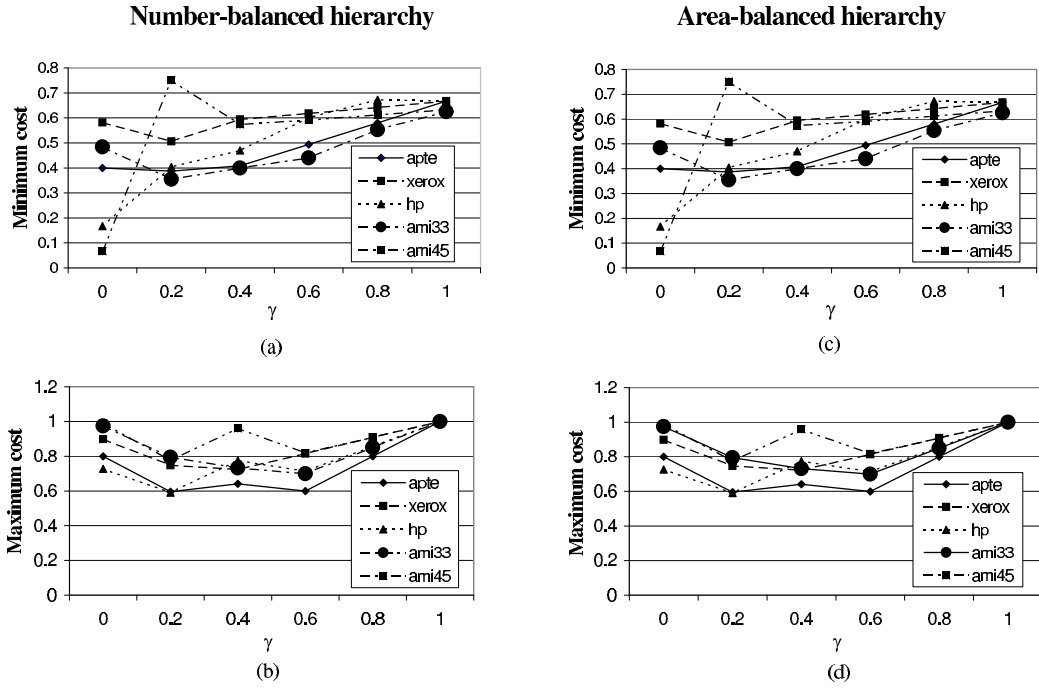


Figure 2.13: Effect of  $\gamma$  on minimum and maximum values of cost over all levels of number and area-balance hierarchy of ms-cuts

on the cost of mixed optimization we have shown the plot of minimum and maximum cost over all levels for all the benchmarks in Figure 2.13. One important observation from these plots is that there is no notable difference between number and area balance bipartition as far as minimum and maximum values of cost are concerned. This signifies that the type of balance does not have a strong impact on optimization cost. Another point to be noted is that for  $\gamma \geq 0.4$ , the cost does not vary much for all the plots and for all the benchmarks. Hence, to get satisfactory results, the heuristic should be operated for that range of  $\gamma$ .

In Figure 2.14 we have concentrated on the 1st level of the hierarchy of cuts as this is the most computation-intensive as well as the most important one since it affects the quality of the cuts in the subsequent levels also. For both number and area-balance type we have plotted the balance-ratios ((a) and

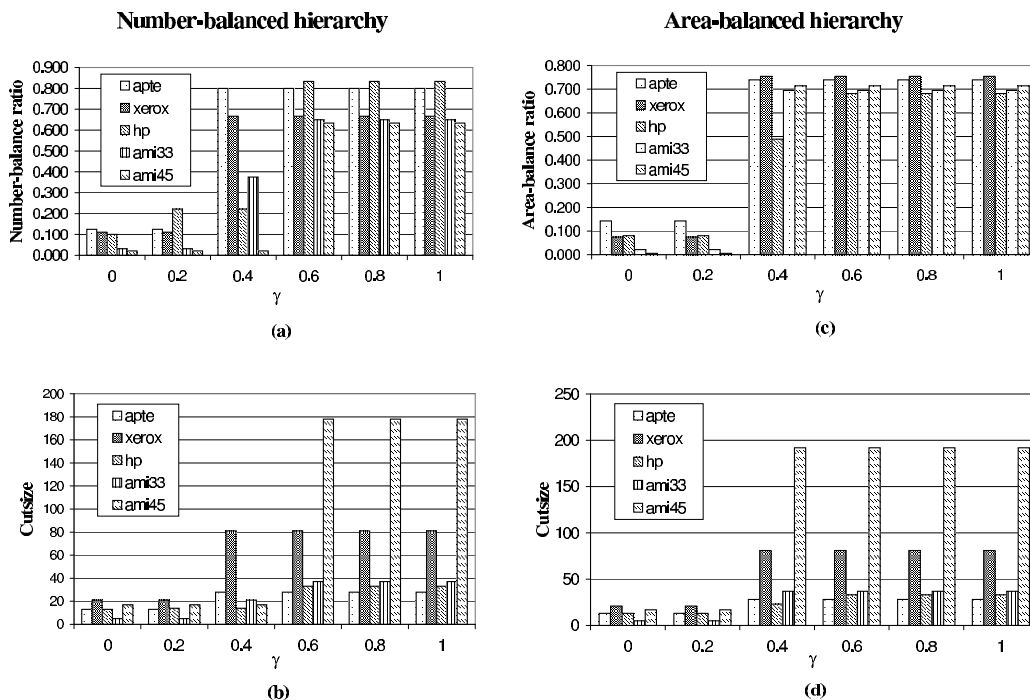


Figure 2.14: First level statistics

(b)) and the cutsizes ((c) and (d)) at the first level for all the benchmarks. We find that for values of  $\gamma \geq 0.4$  the balance factor becomes satisfactory as well as stable for both type of experiments. The cutsize however increases for the range  $\gamma \geq 0.6$ . So we can deduce that the most effective range is  $0.4 \leq \gamma \leq 0.6$ , when the trade-off between cutsize and balance-ratio is optimal.

Finally, Table 2.4 reports the number of levels versus  $\gamma$ . It also gives the maximum cutsize over all levels for different values of  $\gamma$  and the CPU time. The CPU time taken to partition the benchmarks is indeed very low as expected since we have used a max-flow-based method, which is itself a low complexity algorithm. In Figure 2.15, number of levels and maximum cutsize are plotted against  $\gamma$ . We find the number of levels falls sharply ((a) and (b)) in the zone  $\gamma = 0.4$  and then maintains the value up to  $\gamma = 1$ . So for keeping the number of levels and in turn the CPU time low,  $\gamma = 0.4$  is a good engineering choice. Increasing  $\gamma$  further to increase balance factor does not help much. In the

Table 2.4: Variation of number of levels and maximum cutsize with  $\gamma$

Name of benchmark	$\gamma$	Number-balance type			Area-balance type		
		# levels	Max cutsize over all levels	CPU time (in Sec)	# levels	Max cutsize over all levels	CPU time (in Sec)
apte	0	6	13	0.020	6	13	0.010
	0.2	4	15	0.030	4	13	0.010
	0.4	2	28	0.020	2	28	0.020
	0.6	2	28	0.020	2	28	0.020
	0.8	2	28	0.010	2	28	0.010
	1	2	28	0.010	2	28	0.010
xerox	0	7	61	0.050	7	61	0.040
	0.2	6	61	0.060	6	76	0.020
	0.4	2	81	0.050	3	81	0.020
	0.6	2	81	0.030	3	81	0.040
	0.8	2	81	0.020	3	81	0.020
	1	2	81	0.020	3	81	0.010
hp	0	8	16	0.020	8	16	0.020
	0.2	6	16	0.010	8	16	0.020
	0.4	4	30	0.010	4	23	0.020
	0.6	2	33	0.030	2	33	0.020
	0.8	2	33	0.010	2	33	0.030
	1	2	33	0.020	2	33	0.020
ami33	0	25	16	0.080	25	16	0.090
	0.2	18	16	0.180	18	16	0.130
	0.4	5	21	0.040	4	37	0.040
	0.6	4	37	0.040	4	37	0.060
	0.8	4	37	0.030	4	37	0.050
	1	4	37	0.050	4	37	0.050
ami49	0	43	25	0.440	43	25	0.470
	0.2	43	25	0.510	43	24	0.460
	0.4	5	173	0.280	5	192	0.150
	0.6	4	178	0.170	5	192	0.160
	0.8	4	178	0.130	5	192	0.160
	1	4	178	0.190	5	192	0.160

other two plots ((c) and (d)), the maximum cutsize across all levels is plotted against  $\gamma$ . However, we find that the best choice is to keep  $\gamma \leq 0.4$ .

## 2.13 Conclusion

In this chapter, we have considered the problem of determining the mincost staircase channel (MSC) in a VLSI floorplan. The objective is to find a cross-country monotone staircase channel from one corner of the floorplan to its diagonally opposite corner such that the number of distinct nets crossing the channel is minimum. We have considered both the two-terminal and multi-

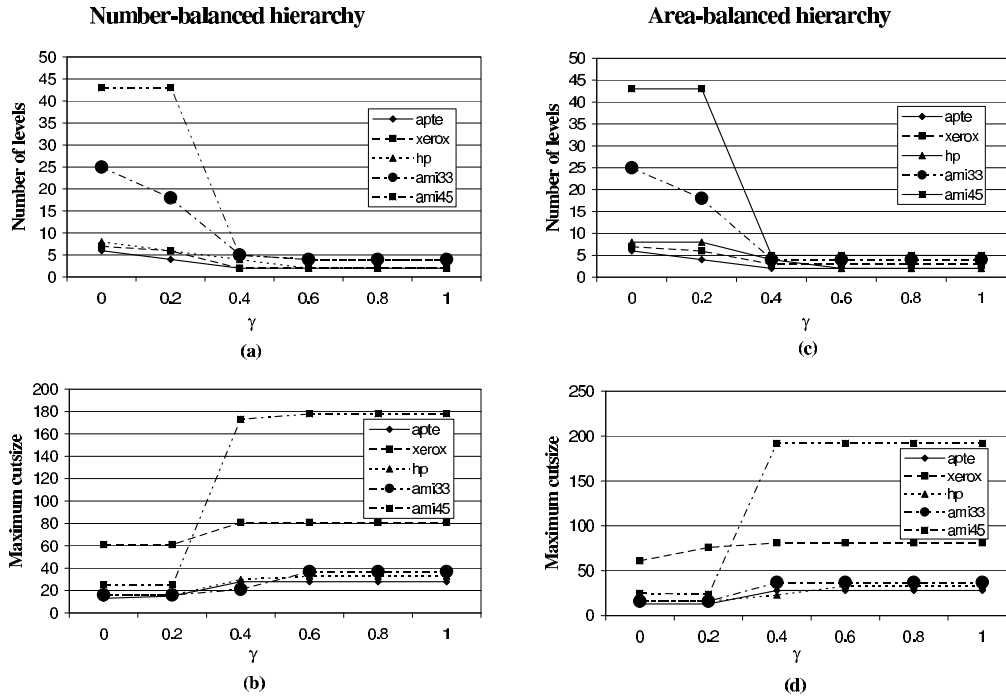


Figure 2.15: Variation of number of levels and maximum cutsizes

terminal net problems. Formulation using max-flow-min-cut problem for a weighted digraph leads to efficient polynomial time algorithms for both of these problems. Hierarchical partitioning of the floor can be accomplished by finding a sequence of staircase channels. To make the hierarchy effective, it is also required the staircase partitions obtained at different level are somewhat balanced also. It is shown that finding a staircase in a floorplan that balances area is NP-complete and hence the mixed optimization problem that simultaneously balances area as well minimizes the number of crossing nets is NP-hard. A flow-based balanced bipartitioning method is designed to generate a hierarchy of monotone staircases where the staircases are chosen with the intention of having as few crossing nets as possible. Our experiments on the benchmark floorplans establish that mixed optimization of balance (number or area) and crossing nets can be achieved efficiently. It has been established that the value of  $\gamma$  should preferably chosen around 0.4 to obtain satisfactory out-

## **Floorplan-driven Partitioning**

---

come on a wide range of testcases. This problem has potential application to minimization of global routing congestion, or repeater placement in the device layer.

# Chapter 3

## Topological Routing

### 3.1 Introduction

The problem of topological routing is reincarnation of global routing in the state-of-the art automated layout synthesis. Topological routing approach has gained importance, particularly for the mixed block and cell design style [112] – a combination of full-custom and standard-cell style. The objective is to generate a *sketch* [69, 33, 32], i.e., a set of lines connecting the terminals of the nets on pre-designed and placed blocks. This sketch is subsequently transformed into a geometrical routing satisfying the design rules [46]. Testing homotopic routability of a given sketch on a single layer is solvable in polynomial time and space [69, 95, 71, 72].

### 3.2 Motivation

The main aim of topological routing is to solve global routing efficiently. In traditional global routers like sequential maze routers or line probe methods [112], nets are routed one by one, usually along the shortest path currently available

## Topological Routing

---

through obstacles. In those cases, the ordering of nets plays a critical role in determining whether 100% routing is achievable. Sometimes, reordering also cannot guarantee completion of routing even though a solution exists. This happens because the routers emphasize on finding the shortest path at every instance. In this context, it is worth mentioning that finding two vertex disjoint paths between two vertices in a planar graph is NP-hard [39].

An alternative approach that may lead to a better solution is to view the problem globally in a concurrent manner, rather than net by net. While there are concurrent global routers primarily based on integer programming formulation, topological routing approach is more appropriate for large problem instances. For routing in the presence of pre-routed special nets or groups of blocks, these pre-routed groups can be represented as polygonal obstacles for the router. Thus the challenge is to route all the nets concurrently with polygonal blocks. Such routing paradigm along with efficient layer assignment is likely to address many of the salient issues in current VLSI layout synthesis.

In this chapter, the problem of topological routing of  $m$  nets  $\{n_1, n_2, \dots, n_m\}$  in the presence of  $k$  non-overlapping *convex polygonal* obstacles  $\{P_1, P_2, \dots, P_k\}$  is addressed. These obstacles represent one or more functional blocks with pin alignment and/or routing of local nets completed. We assume that a net appears not more than once on a polygonal obstacle and the pins of the corresponding nets are placed along the edges of the polygons. The sketch is generated by processing all the nets concurrently in order to circumvent the net ordering problem in traditional global routers mentioned above.

Most of the existing topological routers for multilayer MCMs consider routing space with vias as the only obstacles, whereas we are concerned with routing in the presence of pre-placed finite polygonal obstacles in the routing layers. The associated layer assignment problem is not being considered here.

The method presented here mainly comprises of the following major steps

as shown in the flowchart of Figure 3.1 –

- (i) decomposing the free space available for routing into trapeziums,
- (ii) constructing a routing graph with separation constraints as weights,
- (iii) traversing the graph to find routing paths for all the nets and at the same time satisfying separation constraints.

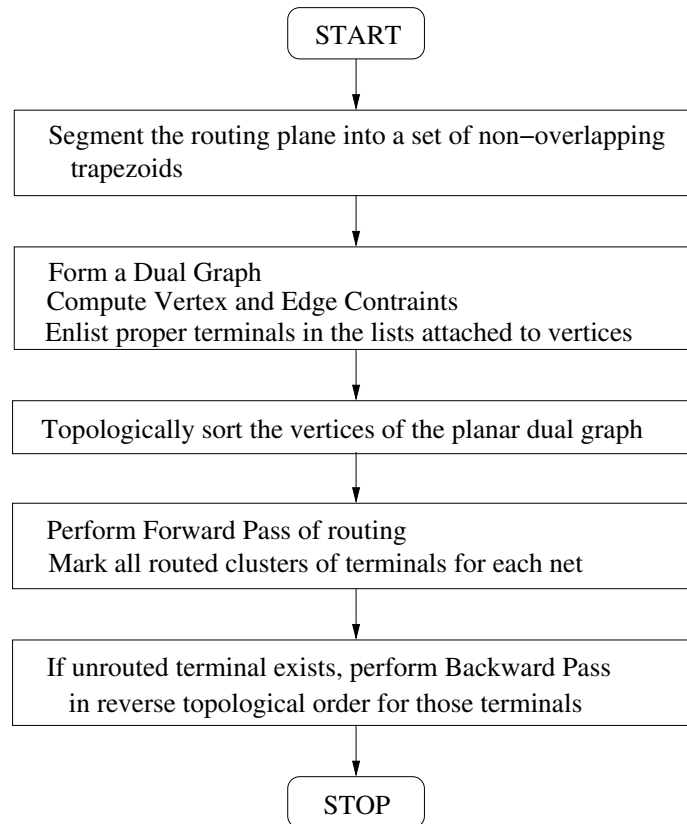


Figure 3.1: Main steps of the algorithm

### 3.3 Trapezoidal decomposition

The first major task is to partition the free intermodular space on the floor into trapeziums in such a manner that the edges of these trapeziums satisfy the minimum separation design rule. We perform a computational geometric horizontal line sweep to obtain this decomposition. Then, a directed graph

## Topological Routing

---

$D$  that is the geometric dual of the planar trapezoidal decomposition, is constructed from it. The following discussion details the major aspects of this module.

**Input** : A rectangular floor  $F$  with convex polygonal obstacles. The obstacles are pre-routed modules with interconnection terminals for a set of nets  $N$  on their boundaries. Without loss of generality, we assume that no polygon touches the boundary of  $F$  and any two terminals on the same polygon belong to distinct nets.

**Output** : A directed acyclic graph where each vertex corresponds to a trapezium and an edge between vertices  $v_i$  and  $v_j$  corresponds to a common trapezoidal edge. The trapeziums are generated by partitioning the free space (rectangular area minus the polygonal area) into a set of mutually exclusive and collectively exhaustive trapeziums.

**Method** : There are two parts in the process -

- (a) generation of trapeziums,
- (b) generation of dual graph from the trapeziums.

**Generation of trapeziums** : We use a horizontal sweep line  $S$  to sweep over the routing space  $F$  from top to bottom. In fact,  $S$  may be moved from any side of the rectangle  $F$  to the opposite one resulting in a different set of trapezoidal partitions. However, the set is unique for any particular direction of sweep. In our implementation, we move the sweep line from the top of  $F$  to its bottom to obtain the trapeziums.

**Implementation Steps** :

1. First, we sort the vertices of the convex polygonal obstacles in descending order of their  $y$ -coordinate values and put them in list  $L_y$ .
2. For each distinct value  $y'$  occurring in  $L_y$  in the sorted order, do

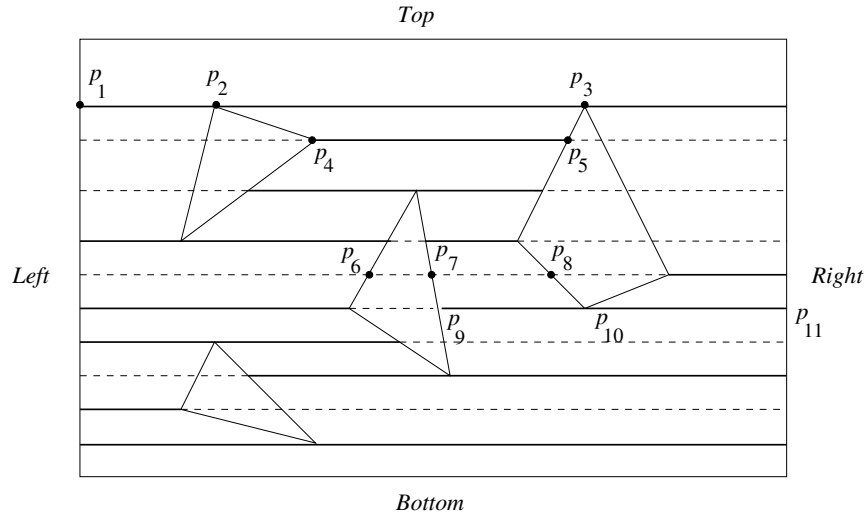


Figure 3.2: Generation of trapeziums in  $F$

- 2.1 Consider that the sweep-line  $S$  is now positioned at an ordinate  $y'$ . Obtain all the vertices of polygons with the same  $y$ -coordinate  $y'$ . Let  $L_v^{y'} = \{x_1, x_2, \dots, x_k\}$  be the  $x$ -coordinates of those polygonal vertices lying on  $S$ .
- 2.2 Obtain all the points of intersection of  $S$  with the set of polygonal edges and the two boundaries *Left* and *Right* of the floor  $F$ . The polygonal edges are kept sorted in descending order of their maximum  $y$ -value. Let these intersection points be  $L_e^{y'} = \{x_m, x_{m+1}, \dots, x_r\}$ .
- 2.3 Merge  $L_v^{y'}$  and  $L_e^{y'}$ , and then sort in ascending order to get the list  $L_x^{y'}$ .
- 2.4 Traverse  $L_x^{y'}$  to identify the list of trapezium edges  $L_{T_e}^{y'}$  at  $y'$  as follows. Two consecutive points in  $L_x^{y'}$  form a *candidate edge*  $c$ . If neither end-point of  $c$  is a polygonal vertex, e.g.,  $(p_7, p_8)$  in Figure 3.2, or both lie on same polygon, e.g.,  $(p_6, p_7)$ , then  $c$  cannot form a side of a maximal trapezium and hence is not allowed to enter  $L_{T_e}^{y'}$ . The remaining candidate edges like  $(p_1, p_2)$ ,  $(p_2, p_3)$ ,  $(p_4, p_5)$  are trapezium edges and are

inserted into  $L_{T_e}^{y'}$ , in the same order as they are considered. All trapezium edges are shown by thick solid lines. Note that a sequence of two or more consecutive candidate edges may together form a trapezium edge, e.g.,  $(p_9, p_{10})$  and  $(p_{10}, p_{11})$  together form the top edge of a valid maximal trapezium.

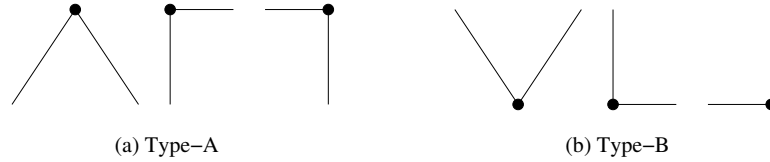


Figure 3.3: Types of end-points for trapezium edges

- 2.5 Traverse the list  $L_{T_e}^{y'}$  to match the top and bottom edges of the trapeziums. The top edge of the first trapezium will be side *Top* of  $F$ . Similarly, the bottom edge of the last trapezium will be side *Bottom* (Figure 3.2) of the rectangular floor. The criteria for identifying the top and bottom edges of the trapeziums are as follows. If for an edge one of the end-points is of type-A (Figure 3.3(a)), that edge cannot be a bottom edge of any trapezium. If an edge is ruled out for being a bottom edge, we consider whether it can combine with the immediately next edge in  $L_{T_e}^{y'}$  to form a bottom edge of some trapezium. Similarly, if one of the end-points is of type-B (Figure 3.3(b)), then that edge cannot be the top edge of any trapezium. However, here also it can combine with another edge to become the top edge of some other trapezium. An edge whose both end-points are neither of type-A nor of type-B, is a top edge for some trapezium as well as a bottom edge for some other trapezium. Also, note that, other than side *Top* (*Bottom*) of floor  $F$ , every top (bottom) edge of any trapezium is either the bottom (top) edge, or a portion of a bottom (top) edge or a combination of several bottom (top) edges of some other trapeziums. To report a trapezium, a

bottom edge has to be matched with a previously detected top edge. A global list  $U_{T_e}$  of unmatched top edges is maintained. It is initialized to side  $Top$  and this is matched with the sole bottom edge generated at the first position of  $S$ . When a new top edge is detected, it is immediately inserted in  $U_{T_e}$  for future matching. As soon as a bottom edge  $e_b$  matches (procedure is being discussed below) with a top edge  $e_t$  in  $U_{T_e}$ , the edge  $e_t$  is deleted from  $U_{T_e}$  and a new trapezium is reported. The reported trapeziums are stored in two arrays  $A$  and  $B$ . Array  $A$  ( $B$ ) contains the set of trapeziums sorted according to the  $y$ -coordinates of their top (bottom) edges.

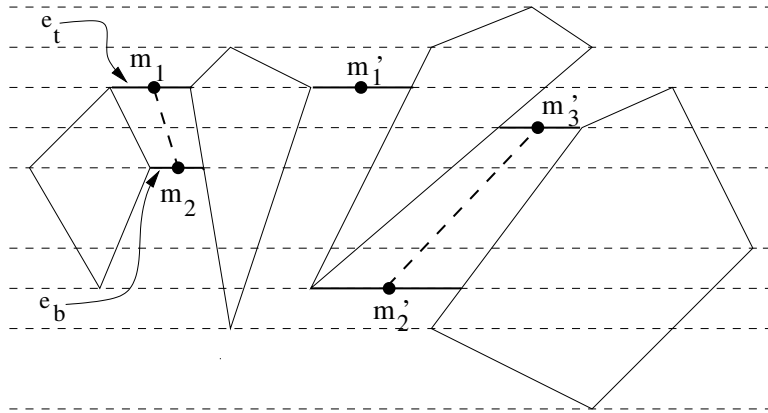


Figure 3.4: Matching of top and bottom edges for a trapezium

**Matching of a top edge with a bottom edge:**

Consider Figure 3.4. Matching  $e_b$  with an  $e_t$  is performed as follows:

- (i) among all edges in  $U_{T_e}$ , consider the edge  $e_t$  whose midpoint is nearest to that of  $e_b$ ;
- (ii) then check whether the left end-points of both the edges lie on the same polygonal edge, if yes, match  $e_t$  with  $e_b$  (e.g.  $m_1$  and  $m_2$ ), if not (e.g.  $m'_1$  and  $m'_2$ ), the top edge with next nearest mid-point is checked (e.g.  $m'_3$  and  $m'_2$ ).

The above steps are repeated for all positions of  $S$ , i.e., all distinct values

## Topological Routing

---

of  $y$ -coordinates. The last position of  $S$  generates only one top edge, which is matched with side *Bottom* of  $F$ .

**Lemma 3.1** *The trapezoidal decomposition algorithm for  $n$  polygonal vertices is  $O(n \log n)$ .*

**Proof.** It follows from the standard sweep line techniques [31]. Note that the number of distinct positions for which we have to consider the sweep-line is  $O(n)$ . Also at any particular position of  $S$ , the processing of each polygonal vertex can be performed in constant time and hence the total work for  $n$  vertices can be performed in  $O(n)$  time. The number of viable trapezium edges is also  $O(n)$  as each viable edge must have at least one end-point coinciding with a polygonal vertex. The total amortized time required for processing all the viable edges is again  $O(n)$ . Hence the time complexity is dominated by the sorting requirements, with overall complexity  $O(n \log n)$ .  $\square$

**Dual Graph Generation :** Consider two arrays of trapeziums  $A$  and  $B$  discussed before.

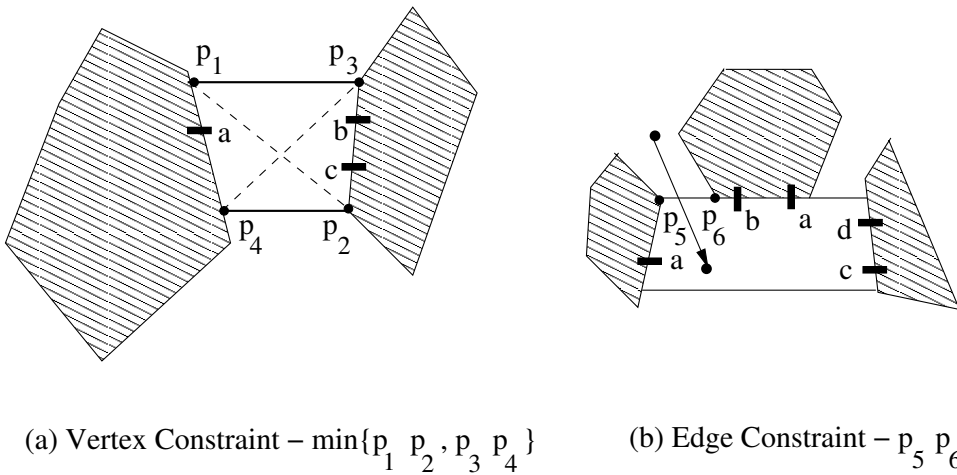


Figure 3.5: Computation of vertex and edge constraints

- For each trapezium  $T$  in  $A$ , create a vertex  $v$  of the directed dual graph  $D$ . The lists of incoming and outgoing edges of  $v$  are initialized and the constraint associated with  $v$  is computed as the minimum of the lengths of the two diagonals of  $T$  (Figure 3.5(a)).
- Each vertex  $v$  has a list  $Orgt(v)$  of original terminals that correspond to the terminals lying on the edges of  $T$ . These terminals of  $v$  are obtained by examining the polygonal edges of  $T$ . It should be noted that any non-horizontal edge of a trapezium corresponds to a unique polygonal edge (Figure 3.5(a)).
- For the bottom edge  $e_b(T)$  of  $T \in A$ , obtain from array  $B$ , the list  $L_T$  of trapeziums with top edges overlapping it, i.e., having same  $y$ -coordinate and intersection of the horizontal intervals. Create a directed edge from  $v$  corresponding to  $T$  to each of the vertices corresponding to trapeziums in  $L_T$ .
- Each edge  $e$  in  $D$  has an associated constraint computed as the minimum of the lengths of the bottom edge and top edge whose overlap is responsible for the existence of  $e$  (Figure 3.5(b)).

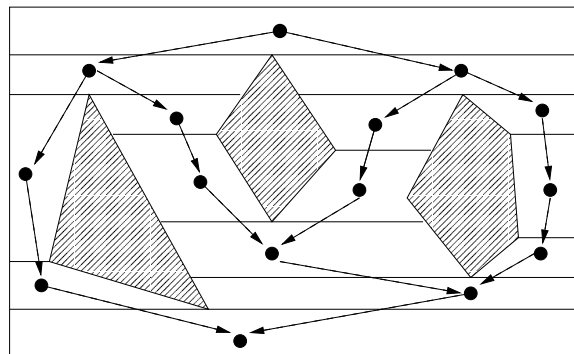


Figure 3.6: Dual graph for a typical example

**Lemma 3.2** *The dual graph  $D$  is planar and acyclic.*

**Proof.** It easily follows from the fact that the input set of non-overlapping trapeziums were embedded in a plane and all edges of  $D$  are directed from top to bottom as shown in Figure 3.6.  $\square$

## 3.4 Routing algorithm

### 3.4.1 Overview

A two-pass greedy heuristic algorithm produces the routing of the nets concurrently by traversing the trapezoidal regions in the line-sweep order, the first (second) pass going from the top (bottom) boundary to the bottom (top) boundary of the layout is called the forward (backward) pass. One pass of the greedy heuristic essentially involves a traversal of the vertices of the graph  $D$  in a topologically sorted order. Initially, the list of nets whose terminals appear on the edges of a trapezium is associated with the corresponding vertex in  $D$ . During the traversal, an unrouted net is passed on from one vertex to all its neighbors downward. We call this as broadcast of a net. At any intermediate stage of a pass, the sweep line touches all the trapeziums at the same horizontal level. If a net  $n_j$  occurs in the netlists of two trapeziums on the sweep line, then  $n_j$  is topologically routed from each of these two trapezoidal regions to their common ancestor  $T$  in  $D$ , if  $T$  has an original terminal of  $n_j$ . Note that even if they do not have a common ancestor, they might get connected through a common successor later on. The nets which are left unrouted in the forward pass are considered again during the next pass where the sweep is in the reverse direction.

Local congestion is modeled by assigning a capacity constraint to the edges of the trapeziums. At any stage of the routing algorithm, a net is passed on from one trapezium to its neighbor, only if the net-density of the edge between

the two trapeziums is less than its capacity. However, capacity constraints only on the trapezoidal edges may be inadequate in modeling congestion. There are counter-examples where capacities of some additional cutlines have to be included. So we have introduced the concept of shortest diagonal to take into account the smallest constriction within a trapezium.

### 3.4.2 Algorithm for routing by traversal of dual graph

**Input:** A directed graph  $D = (V, E)$  which is the dual graph resulting from the trapezoidal decomposition, each vertex  $v$  having a list  $Orgt(v)$  of terminals.

**Output:** Global routing path for each net respecting the constraints on edges and vertices of  $D$ .

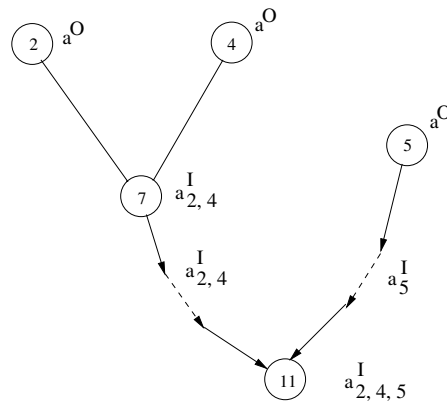


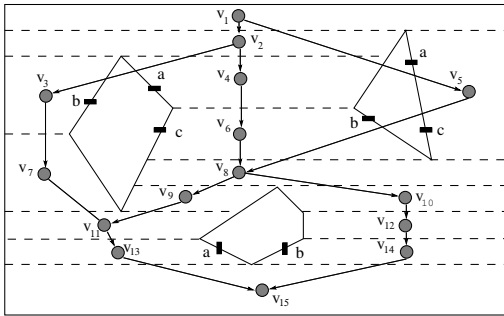
Figure 3.7: Original and inherited terminal lists for net  $a$

**Data Structure:**

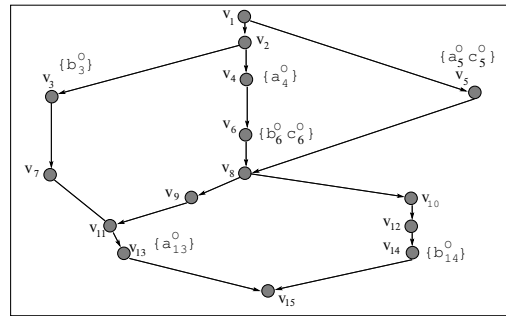
With each vertex  $v$ , two lists of terminals are associated,  $Orgt(v)$  and  $Inht(v)$ . The former is an input whereas the latter one gets constructed as the algorithm proceeds.  $Inht(v)$  contains the terminals of nets that  $v$  inherits from its predecessors as the traversal continues. For each inherited terminal  $t$ , (for net  $a$  say) there is a list  $route(t)$  consisting of those vertices having original terminals of  $a$ . If there are more than one entries in  $route(t)$ , then it means

## Topological Routing

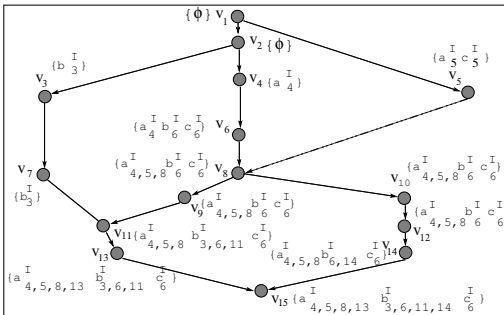
the original terminals of  $a$  in all those vertices are already connected to each other by some path. A terminal in the list  $Orgt(v)$  ( $Inht(v)$ ) of vertex  $v$  is indicated with superscript  $O$  ( $I$ ). Consider Figure 3.7. There are three original terminals of net  $a$  in vertices  $v_2$ ,  $v_4$  and  $v_5$ . There exists a routing path between the terminals of  $v_2$  and  $v_4$  through  $v_7$  and we denote the inherited terminal of net  $a$  in  $v_7$  as  $a_{2,4}^I$ . Also  $v_2$  and  $v_4$  will be present in  $route(a_{2,4}^I)$ .



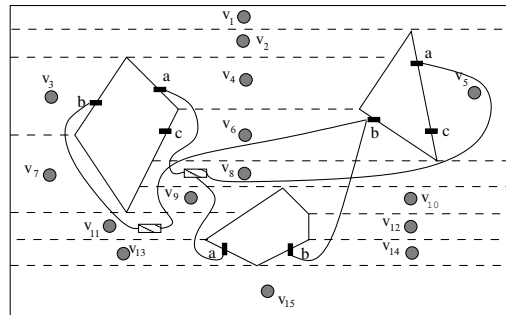
a) Trapezoidal Decomposition and Formation of Dual Graph



b) Dual Graph with Original Terminals



c) Dual Graph with Inherited Terminals



d) After Topological Routing

Figure 3.8: An example of broadcasting and routing in the forward pass

**Method:** In the forward pass the traversal is performed by considering the vertices of the dual graph  $D$  in topologically sorted order. The terminals, both original and inherited, in each vertex is broadcast to all its successors. On reaching the successor, all these terminals are then considered as inherited terminals. Note that in the forward pass, the actual routing path is determined in reverse traversal mode. This should not be confused with backward pass.

In the backward pass, the traversal is performed by considering the vertices in reverse topological order, whereas the actual routing is done in forward traversal mode. In the typical example of Figure 3.8, nets  $a$  and  $b$  got successfully routed in the forward pass, however net  $c$  could not be routed due to constraint violation in edge  $(v_5, v_8)$ , shown as a dotted one in Figure 3.8(c). It however gets routed in the backward pass as shown in Figure 3.9. In Figure 3.8(d), actual routing paths are shown using curved lines. Also in those vertices, where a particular net does not have any original terminal, a Steiner point for that net is shown as a patterned box. Note that if we consider the interconnected routed wires of a net as a Steiner tree, then the original terminals will be the demand points and all the other points where two or more wires meet can be called as Steiner points [113]. Below we list down the detailed steps of the main algorithm, which uses a procedure named *routeBack* presented afterward. The text appearing after the '//' sign are comments as usual.

**Steps:**

1. Ordering: Topologically sort vertex set  $V = \{v_1, v_2 \dots v_n\}$  of digraph  $D$  using DFS.
2. Forward Pass: For each vertex  $v_i$  in the topological order do -
  - For each net  $a$  in arbitrary order do -
    - if  $Orgt(v_i)$  contains  $a_i^O$ , then put  $a_i^I$  in  $Inht(v_i)$ ;  
 // all inherited term. of predecessors are already present in  $Inht(v_i)$
    - while there exists more than one terminal of  $a$  in  $Inht(v_i)$  do -  
 // process the terminals two at a time  
 // consider subcases to identify if  $v_i$  is on the routing path of net  $a$   
 // inherited terminal may be single term. or partially routed clusters  
 // eg. given below within comments for diff. cases refer to Fig. 3.8(c)
      - \* Case A:  $Inht(v_i)$  has  $a_i^I$  and a single terminal  $a_k^I$  for an original terminal in  $v_k$  -  
 call  $routeBack(v_i, a, a_i^O, a_k^I)$  to find a path from  $v_i$  to  $v_k$ ;  
 if  $returnValue$  is  $status0$ ,  
 then remove  $a_i^I$  and  $a_k^I$  from  $Inht(v_i)$  and add  $a_{i,k}^I$ ;  
 //  $b_{14}^I$  and  $b_6^I$  forms  $b_{6,14}^I$  in  $v_{14}$   
 else remove  $a_k^I$  from  $Inht(v_i)$ ;

## Topological Routing

---

- \* Case B:  $Inht(v_i)$  has two distinct single terminals  $a_j^I$  and  $a_k^I$ , ( $j, k \neq i$ ) -  
 call  $routBack(v_i, a, a_j^I, a_k^I)$  to find a path from  $v_j$  to  $v_k$  making  $v_i$  a Steiner point;  
 if *return Value* is *status0*,  
     then remove  $a_j^I$  and  $a_k^I$  from  $Inht(v_i)$  and add  $a_{i,j,k}^I$ ;  
     //  $a_4^I, a_5^I$  forms  $a_{4,5,8}^I$  in  $v_8$   
 else remove  $a_j^I$  (for *status1*) or  $a_k^I$  (*status2*) or both (*status3*);  
 //  $c_5^I$  removed in  $v_8$
  - \* Case C:  $Inht(v_i)$  has  $a_i^I$  and a cluster  $a_{p,q}^I, p, q \neq i$  -  
 call  $routBack(v_i, a, a_i^O, a_{p,q}^I)$  to find a path from  $v_i$  to a Steiner pt. in any predecessor of  $v_i$ ;  
 if *return Value* is *status0*,  
     then remove  $a_i^I$  and  $a_{p,q}^I$  from  $Inht(v_i)$  and add  $a_{i,p,q}^I$ ;  
     // follow net  $a$  in  $v_{13}$   
 else remove  $a_{p,q}^I$  from  $Inht(v_i)$ ;
  - \* Case D:  $Inht(v_i)$  has one single terminal  $a_j^I, j \neq i$  and a cluster  $a_{p,q}^I, p, q \neq j$  -  
 call  $routBack(v_i, a, a_j^I, a_{p,q}^I)$  to find a path from an original terminal in  $v_j$  to a Steiner point;  
 if *return Value* is *status0*  
     then remove  $a_j^I$  and  $a_{p,q}^I$  from  $Inht(v_i)$  and add  $a_{i,j,p,q}^I$ ;  
 else remove any one of them or both as per the status returned;
  - \* Case E:  $Inht(v_i)$  has two non-intersecting clusters  $a_{p,q}^I$  and  $a_{x,y,z}^I$  -  
 Call  $routBack(v_i, a, a_{p,q}^I, a_{x,y,z}^I)$  to find a path between two Steiner points;  
 if *return Value* is *status0*,  
     then remove  $a_{p,q}^I$  and  $a_{x,y,z}^I$  from  $Inht(v_i)$  and add  $a_{i,p,q,x,y,z}^I$ ;  
 else remove any one of them or both as per the status returned;
  - \* Case F:  $Inht(v_i)$  has two intersecting clusters  $a_{p,q}^I$  and  $a_{q,r}^I$  -  
 // no need to route as they are already connected through  $v_q$   
 remove  $a_{p,q}^I$  and  $a_{q,r}^I$  from  $Inht(v_i)$  and add  $a_{p,q,r}^I$ ;  
 // follow net  $a$  in  $v_{15}$
- For each successor  $v_j$  of  $v_i$  in  $D$ , put all the terminals of  $Inht(v_i)$  into  $Inht(v_j)$ ;

3. Backward Pass: Process only those nets that are still unrouted. Proceed exactly as STEP 2 but process the vertices in reverse topological order (Figure 3.9). Also pass inherited terminals to the predecessors instead of successors in  $D$ .

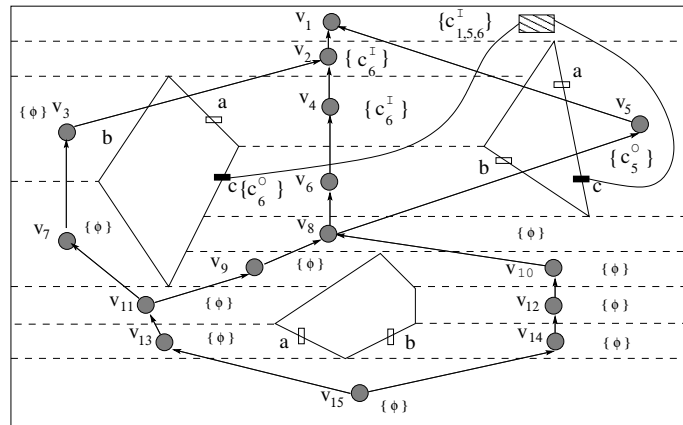


Figure 3.9: An example of broadcasting and routing in the backward pass

The routing of the net  $c$  that could not be routed in the forward pass of the example of Figure 3.8, is being shown in Figure 3.9 in a consolidated manner.

The following procedure performs the actual routing using backtracking.

**procedure** *routeBack*(vertex  $v_i$ , net  $a$ , terminal  $t_1$ , terminal  $t_2$ )

**Output:** If successful, a routing path for the net  $a$  from  $t_1$  to  $t_2$ . The routing path has to meet the vertex and edge constraints.

**Boolean flag:** *success1*, *success2*

**Steps:**

1.  $\text{currentVertex} \leftarrow v_i$ ;
2. if ( $t_1$  is an original terminal or a Steiner point in  $\text{currentVertex}$ ) do
3.     Set *success1* True; // routing from  $t_1$  up to a point in  $v_i$  is successful
4.     goto Step 14;
5. else
6.     Let  $V'$  be the set of predecessor vertices of  $\text{currentVertex}$  that has terminal  $t_1$  of net  $a$ ;
7.     while ( $v_j \in V'$  is unmarked) do
8.         Mark  $v_j$ ;

## Topological Routing

---

9.       if (both edge  $(v_j, v_i)$  and  $v_j$  meet the routing constraints) do
10.       Put  $(v_j, v_i)$  in a temporary route path of net  $a$ ;
11.       currentVertex  $\leftarrow v_j$ ;
12.       goto Step 2;
13.    Set *success1* False; // If we reach here, routing is unsuccessful for  $t_1$
14. currentVertex  $\leftarrow v_i$ ;
15. Process  $t_2$  exactly like  $t_1$  and likewise set *success2* True or False;
16. if *success1* do
17.    if *success2* do
18.       Put the edges from temporary to actual route path of net  $a$ ;
19.       return *status0*; //  $t_1, t_2$  can be successfully connected through  $v_i$
20.    else
21.       return *status2*; // terminal  $t_2$  had routing problems
22. else
22.    if *success2* do
23.       return *status1*; // terminal  $t_1$  had routing problems
24.    else
25.       return *status3*; // both the terminals had routing problems

**Theorem 3.1** *Each pass of the above greedy method of topological routing runs in  $O(|E|.|N| + |V|.|T|)$  time where  $D = (V, E)$  is the dual graph,  $T$  is the set of terminals and  $N$  is the set of nets.*

**Proof.** The graph  $D$  is planar. In a single pass, for each net any edge is traversed only once during broadcast and once more during backtrack. Also note that per net only one terminal (may be single or a cluster) passes to the inherited terminal list of its successors (predecessors) in the forward (backward) pass. Hence the term  $|E|.|N|$  comes in the time complexity. Also as

the sweep-line passes through, at any vertex the total number of terminals to be processed considering all nets, is always bounded by the number of original terminals on the whole floor. That explains the occurrence of the other term. An amortization may decrease the complexity of this latter term to some extent. □

The method being greedy in nature, the optimality however cannot be provably guaranteed.

## **3.5 Implementation of our algorithm**

Since standard benchmarks for this problem are not known to exist, floorplans containing polygonal obstacles are generated randomly to test our algorithm. First, we briefly outline the method followed to generate the problem instances for our algorithm.

### **3.5.1 Generation of a polygon within a given rectangle**

Generation of a convex polygon within a closed rectangular area involves two major steps. Firstly, we generate the required number  $n$  of non-overlapping rectangles, and then within each of those rectangles, we generate a convex polygon. In the second step, a number of points are generated randomly within a rectangle, and then the convex hull of those points are found out. Following are the elaborated steps of the algorithm:

**procedure genRect( $n$ )** // Input  $n = \#$  of disjoint rectangles

1. for  $i$  in 1 to  $n$  do
2.     produce vertices for new rectangle  $R_i$  within boundary

## Topological Routing

---

- limits of the floor respecting the size constraints;
3. for each  $R$  in global rectangle array  $A_R$  do
  4.     if ( $R$  overlaps with  $R_i$ )
  5.         then GOTO Step 2;
  6.     add  $R_i$  to  $A_R$ ;

### **procedure genPoly()**

1. for each  $R_i$  in  $A_R$  do
2.     generate a set of points  $Q$  within boundary of  $R_i$ ;
3.     find the convex hull  $P$  of  $Q$ ;
4.     add  $P_i$  to global polygon array  $A_P$ ;

The quick rejection algorithm is used to determine the overlapping rectangles and the convex hull is found using Graham's Scan method [31].

### **3.5.2 Random generation of terminals on polygons**

After the polygon generation is complete, the global polygon array  $GP$  is iterated and terminals or net-points are put (generated) on each polygon according to a minimum bound, currently it is 2. The net-points are generated such that, no two net-points on the same polygon come from the same net. The algorithm is given below.

### **procedure genNetPt()**

1. for each polygon  $P$  in  $GP$  do
2.     for (random number of times  $\geq 2$ ) do

3.       get random net  $N$ ;
4.       generate net-point  $N_p$  of  $N$  on  $P$ ;
5.       if ( $N$  not already in  $P$ )
6.             then put  $N_p$  on  $P$ ;
7.    if (number of net-points on  $P < 2$ )
8.       while (number of net-points on  $P < 2$ ) do
9.             get random net  $N$ ;
10.          generate net-point  $N_p$  of  $N$  on  $P$ ;
11.          if ( $N$  not already in  $P$ )
12.             then put  $N_p$  on  $P$ ;

### 3.5.3 Experimental results

Our algorithm yields a valid solution very fast for most examples of moderate size in only two passes – one top-to-bottom sweep followed by one bottom-to-top sweep. The paths generated are obstacle-free but does not necessarily have the shortest distance. Thus this simple heuristic produces solution to the topological routing problem avoiding routing deadlocks caused by shortest path criterion or net ordering in sequential approaches. Table 3.1 presents the performance of our algorithm on a set of randomly generated testcases of moderate size.

## 3.6 Concluding remarks

We have tackled the problem of topological routing of all the nets simultaneously, instead of one Steiner tree after another. The routing was done in the presence of pre-routed nets or groups of circuit blocks. An efficient graph

## Topological Routing

---

Table 3.1: Experimental results

# Polygons	# Nets	# Terminals	Floor Size	Success %	CPU Time (sec)
10	5	32	400 x 300	100.00	.110
10	10	53	400 x 300	100.00	.100
10	20	116	400 x 300	95.00	.200
10	20	123	500 x 400	100.00	.210
20	10	87	400 x 300	100.00	.270
20	20	161	400 x 300	60.00	.430
20	20	170	500 x 400	95.00	.690
20	40	288	400 x 300	77.50	1.110
20	40	295	500 x 400	77.50	.960
20	40	303	700 x 500	95.00	1.490
30	15	136	400 x 300	80.00	.880
30	15	143	500 x 400	86.67	.880
30	15	130	700 x 500	73.33	.640
30	30	240	400 x 300	100.00	1.180
30	60	448	400 x 300	43.33	2.110
30	60	436	700 x 500	91.67	2.480
40	20	186	400 x 300	85.00	1.270
40	20	193	500 x 400	100.00	1.180
40	40	323	400 x 300	80.00	1.750
40	40	341	700 x 500	95.00	4.070
40	40	339	800 x 700	100.00	2.000
40	80	599	400 x 300	51.25	2.560
40	80	606	700 x 500	76.25	2.680
40	80	626	1200 x 1000	95.00	5.880
50	25	231	400 x 300	48.00	2.160
50	25	232	700 x 500	76.00	1.630
50	25	237	800 x 700	100.00	1.660
50	50	411	400 x 300	54.00	2.500
50	50	396	700 x 500	82.00	3.620
50	50	406	1200 x 1000	100.00	3.470
50	100	741	400 x 300	40.00	3.990
50	100	751	700 x 500	68.00	5.550
50	100	719	1200 x 1000	94.00	53.120

traversal based greedy heuristic is proposed. The results obtained from the implementation are very encouraging. Finding an algorithm for layer assignment of the global routing paths for the nets is in the agenda for future work.

# Chapter 4

## Reducing Intersections for Rectilinear Steiner Trees

### 4.1 Introduction

In layout synthesis of a VLSI chip, circuit blocks are placed within a rectangular floorplan boundary and in most of the cases, these blocks are rectangular in shape and are placed isothetically. To each block, a set of terminals belonging to different nets is attached. All the terminals belonging to the same net are to be connected electrically by wire routing. Routing is generally completed in two phases - global routing and detailed routing. The global routing phase first generates a loose route for each net and then the actual geometric layout of each net gets fixed in the detailed routing phase. For most chips, minimizing the total wire-length is the main criterion based on which the routing phases are carried out. However, for high performance chips it is important to route each net such that all the signals meet their timing constraints. In such cases one of the targets may be minimizing the maximum wire-length. The objective of global routing may be different depending upon the design style,

whether the chip follows a full-custom, standard cell or a gate array type of design style [113]. Minimizing the number of vias is a common objective for a number of global or detailed routers, as they are expensive. Vias are used when a wire needs to go from one routing layer to another to avoid getting shorted with another wire belonging to a different net. Increase in number of vias may decrease the yield, as they involve processing of multiple layers. They also introduce parasitic capacitance, which in turn may affect the speed of the chip. In this chapter, we propose a routing strategy that may help in reducing the number of vias used.

## 4.2 Background of the problem

For each distinct net  $n$ , the primary goal of the routing phase is to interconnect all the terminals of  $n$  using minimum length of wire. This is similar to the problem of minimization of total cost of edges in a Steiner tree [113]. The terminals of the net  $n$  are considered as the demand points. The underlying grid graph is the graph defined by the intersections of horizontal and vertical lines drawn through the demand points. It is also known as the Hanan grid. All the other grid-points that are not demand points then become candidates for Steiner points. A Steiner tree whose edges are constrained to rectilinear shapes is called a Rectilinear Steiner Tree (RST). A Rectilinear Steiner Minimum Tree (RSMT) is an RST with minimum cost among all RSTs. So the optimal routing of a multi-terminal net in the manhattan model turns out to be finding the RSMT, which is a computationally hard problem. Hwang proved the following theorem [53].

**Theorem 4.1** *Let  $Cost_{MST}$  and  $Cost_{RSMT}$  be the costs of a minimum cost rectilinear spanning tree and rectilinear Steiner minimum tree, respectively.*

*Then  $\frac{Cost_{MST}}{Cost_{RSMT}} \leq \frac{3}{2}$ .*

As a result, many heuristic algorithms use Minimum Spanning Tree (MST) as a starting point and apply local modifications to obtain an RST. Hence, they can guarantee that weight of their RST is at most 1.5 times the weight of the optimal tree [49, 54]. Actually, the edges should be selected so that they have the maximum overlap amongst themselves, which in turn will minimize the total length (cost) of the tree. In this chapter, we also follow a similar approach, whenever we need to connect multiple terminals of the same net. Given the demand points, we first draw the geometric MST and then convert it to the rectilinear version. We start from there and make a few local improvements to generate some RST in our algorithm. In recent literature, Steiner tree generation has also received considerable attention [128].

Minimizing total wire-length is however not the only goal for routing of multi-terminal nets. Suppose we have to route  $k$  multi-terminal nets on the plane and we mark them with  $k$  distinct colors. In this chapter, we give a strategy for routing these  $k$  nets in such a way that the edge-crossings between the trees of different colors is minimized. Each crossing between two edges of different colors leads to an use of a via so that the corresponding nets get routed through different metal layers, otherwise two distinct nets will get electrically shorted. Hence the rationale behind such a strategy is that it may reduce the number of expensive vias, which may in turn lead to better utilization of the metal layers. Reducing the number of intersections for two or more graphs is also an well-studied subject in the literature [126, 57]. However in order to solve this problem, we first focus on an underlying theoretical problem of monochromatic partitioning. We then develop a fast heuristic  $H_f$  for solving it, with supporting experimental results on randomly generated problem instances. Subsequently we utilize this heuristic to give another novel heuristic  $H_s$  to solve the actual problem.

## 4.3 Monochromatic partitioning

### 4.3.1 Prior work

The problem of partitioning a set of colored points into monochromatic parts has received considerable attention in the recent literature of computational geometry [37, 78]. Some of its variants have nice applications in the design of fault-tolerant multi-modal sensor fusion for embedded sensor networks [13, 16, 61]. Further, this problem has relevance to many other problems including signal-space partitioning required for reliable high-rate digital transmission in pulse-modulation systems [78], and also in the design of rectilinear Steiner trees for multiple nets with fewest intersections [90], which is in fact the topic of this discussion.

Let us first consider the simple problem where a set of points with two different colors is scattered on a plane. Now a set of straight lines of minimum cardinality has to be drawn in such a way that in each cell, bounded or unbounded, there may exist points of at most one color. Lu et al. [78] designated this as the Minimum Linear Classification (MLC) problem. They prove the computational hardness of this problem by a reduction from the Minimum Line Fitting (MLF) problem. Here, we will study a restricted version of the MLC problem, where the lines that can be used to separate the points are all axis-parallel. We show that even this restricted version is NP-complete by a direct reduction from the Vertex Cover problem, which is NP-complete [39]. Calinescu et al. [16] studied the point separation problem. It separates a set of points in the plane no two of which have the same  $x$  or  $y$ -coordinate using minimum number of axis-parallel lines such that each cell of the subdivision contains at most one point. They show that the above problem is NP-complete by a reduction from 3-SAT problem. They have argued that if all the points are colored with separate colors then the problem of separating

the points into monochromatic cells is equivalent to their problem. Thus when the number of colors is part of the input, the colored separation problem is also NP-complete. They further claim that the reduction process used to prove the NP-completeness of point separation problem may be used to prove the hardness of colored version of the separation problem for  $k \geq 2$  colors. However, the proof that we provide by showing the reduction from Vertex Cover, is relatively simple and direct. They suggest a 2-approximation algorithm for their problem by adopting another Linear Programming (LP)-based 2-approximation algorithm due to Gaur et al. [40]. This LP-based scheme was originally used to solve the problem of finding out the minimum number of axis-parallel lines required to hit a set of axis-parallel rectangles on the plane. Incidentally this LP-based technique can be used to give a 2-approximation algorithm for our problem also and in fact, we have compared our results with this technique only. We primarily deal with the following problem. Henceforth, we shall refer to this problem as *minimum linear classification with isothetic lines* or *MILC* problem, in short.

**MILC problem:** Given a set of points of two different colors, scattered on a plane. Draw the minimum number of isothetic straight lines to separate these points into monochromatic cells, bounded or unbounded. This optimization problem may be restated as the following decision problem – does there exist a set of isothetic lines of size  $L$  that separates the points into monochromatic cells?

For simplicity, we may assume that no two points are on the same horizontal or vertical line. However, it makes little difference as far as the solution is concerned. The only point to be noted in this regard is that if two points of different color appear on the same horizontal (resp. vertical) line, then it is mandatory that they have to be separated by some vertical (resp. horizontal) line.

### 4.3.2 Some important observations

We first mention that the one-dimensional version of the *MILC* problem has a very easy  $O(n \log n)$  solution. Here, the points need to be separated into monochromatic bands by drawing parallel straight lines along only one axis. It turns out to be the same as finding the clique-cover for an interval graph generated from the bi-color points.

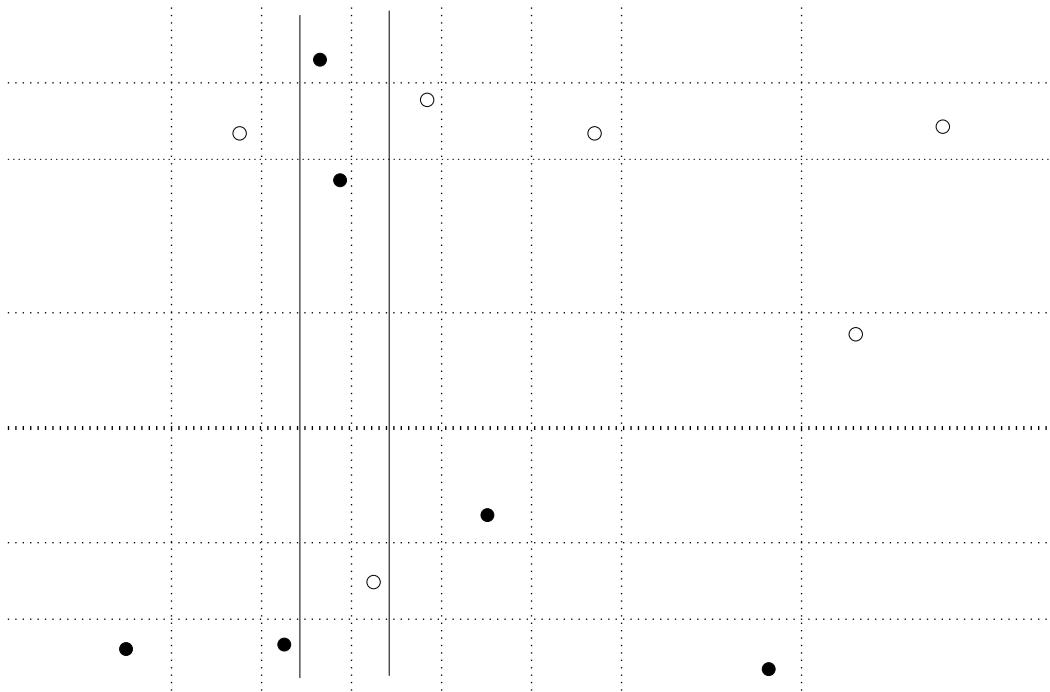


Figure 4.1: A counterexample for monochromatic bands

We show here that a simple conjecture about the *MILC* problem, which appears to be quite obvious intuitively, is actually not true when we give it a closer look. The question is - what are the possible positions through which the required isothetic straight lines can be drawn to satisfy our objective or in other words can we have a finite number of straight lines which will act as the universe from which we can choose the final solution of minimum cardinality. At first, it appears that if we can divide the bi-color points in monochromatic

## Reducing Intersections for Rectilinear Steiner Trees

---

bands by using a set  $H$  of horizontal lines as well as a set  $V$  of vertical lines then the final solution of straight lines should be a subset of  $V + H$ . But in Figure 4.1, we show a counterexample where three straight lines, two vertical and one horizontal, optimally separate the bicolor points into monochromatic cells. The horizontal line belongs to the set  $H$ . However, the two vertical lines do not belong to the set  $V$ . In the figure, the dotted lines are those that separate the bi-color points into monochromatic bands. In the horizontal direction there are 6 such lines and in the vertical direction there are 7 such lines. Hence by our previous discussion, they are the lines belonging to  $H$  and  $V$  respectively. The only horizontal line, which belongs to the optimal solution set is being thickened to distinguish from the other dotted lines. However the vertical lines belonging to the solution set are shown as solid lines as their positions are different from the dotted lines. These two vertical lines actually penetrate through two of the monochromatic bands of points in order to give a solution of minimum cardinality. For this example, if we had restricted ourselves to choose the solution from the dotted lines only, then the cardinality of the solution set would have been more than 3. However, if we are ready to relax the size of the universe a bit, it is easy to find an exhaustive set of straight lines from which we can choose the final solution.

Thus, if the total number of points is  $n$  (considering all possible colors), we consider  $n - 1$  horizontal lines and  $n - 1$  vertical lines as follows. We sort the points with respect to their  $y$  (resp.  $x$ ) coordinates, and place a horizontal (resp. vertical) line between every two consecutive points in that order. These lines serve as the universe of lines, among which we need to find the desired lines for the optimum solution.

### 4.3.3 NP-completeness results

In this section, we first show that the *vertex cover problem* reduces to our proposed *MILC* problem in polynomial time. We take an instance of the vertex cover problem. Let  $G = (V, E)$  be a graph whose minimum sized vertex cover consists of  $k$  vertices. From  $G$ , we generate an instance  $I$  of the *MILC* problem, such that there exists an optimum set of axis-parallel straight lines of cardinality  $(k + |E| + (|V| + 1) + (|E| + 1))$  that separates the bi-colored points of  $I$  into monochromatic cells.

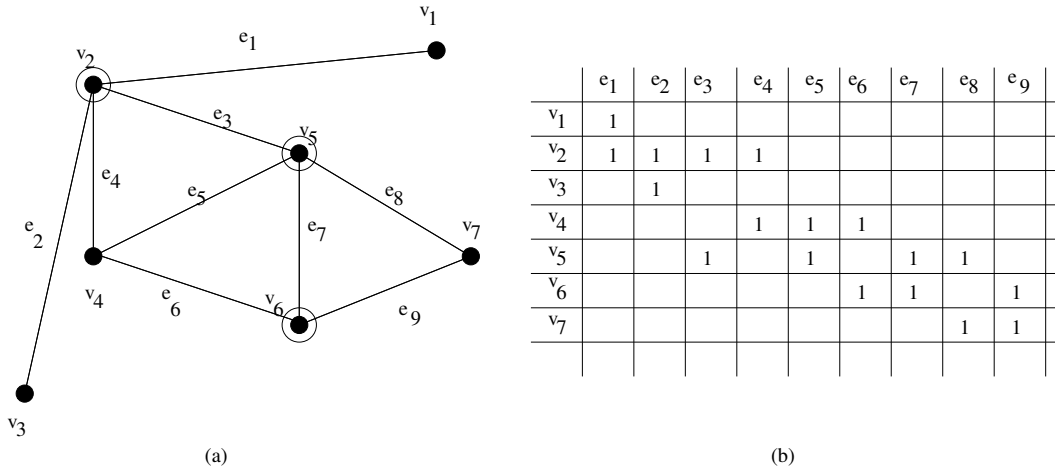


Figure 4.2: An example graph for the proof of NP-completeness

Consider the graph  $G$  shown in Figure 4.2(a). The cardinality of its minimum vertex cover is 3. The highlighted vertices  $\{v_2, v_5, v_6\}$  is one such set. We construct a 0 – 1 matrix  $M$ , whose rows correspond to the vertices and columns correspond to the edges of  $G$ . A "1" in the  $(i, j)$ -th cell indicates that the edge  $j$  is incident on the vertex  $i$  (see Figure 4.2(b)). The other entries of the matrix are "0", which are not shown in the figure. Thus, the vertex cover problem for the graph  $G$  turns out to be the same as identifying minimum number of rows  $R$  to cover all the columns of the matrix  $M$ . We say that a column  $c$  is covered if there exists a row  $r \in R$  such that  $M[r, c] = 1$ .

## Reducing Intersections for Rectilinear Steiner Trees

---

Using this matrix we now construct the corresponding instance of the *MILC* problem, as shown in Figure 4.3.

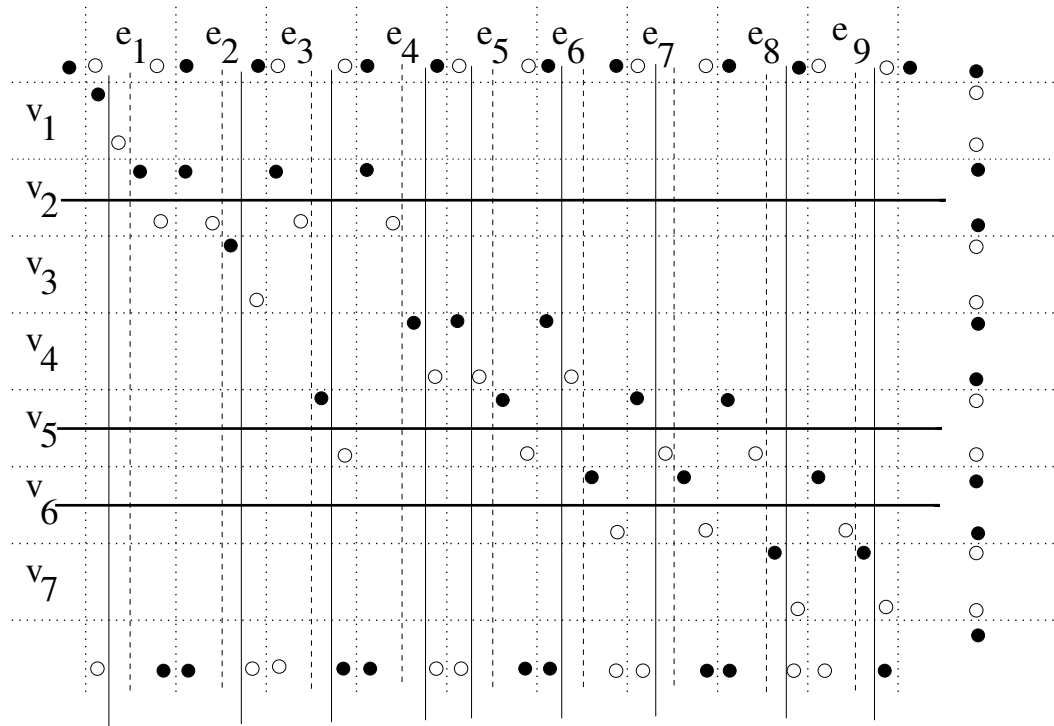


Figure 4.3: Reduction from vertex cover

For each "1" in  $M$ , we put a pair of points (one black and one white) on the plane. We keep the geometric position of this pair relatively same as the position of its corresponding "1" in  $M$ . The motivation of doing so, is to create an instance such that the minimum number of axis-parallel lines required to separate all these pairs of points into monochromatic cells is same as or in some way has a one-to-one correspondence to the minimum number of vertices required to cover all the edges of the graph. To keep the pair of points corresponding to a cell of  $M$  separate from the pair of points corresponding to the other cells, we deliberately add some extra pairs of bi-colored points on the plane. Any routine that will try to separate all the points on the plane into monochromatic cells will also have to separate these extra pairs. To separate

these extra pairs, any routine will need some extra lines and we place these extra points in such a way that those extra separating lines will separate the whole region creating rectangular cells in exact correspondence to the matrix  $M$ . In Figure 4.3, for each row (column) separator of  $M$ , we use a pair of black and white points placed side by side having the same  $x$  ( $y$ )-coordinate at the right (top) of the figure. The dotted lines in the figure are the ones that will be produced by any algorithm trying to separate these extra black and white pairs. It is these dotted lines that will tessellate the plane into  $N$  cells of finite size where  $N$  is the number of cells present in the matrix  $M$ . Note that it is not mandatory that the extra pair of points that we put has to have matching  $x$  ( $y$ )-coordinate for the two points within the pair. It can always be formalized by adding some more points such that the separating line is produced in the same place by any algorithm. Actually we need the separating lines to create a geometry corresponding to the matrix, the number of extra points does not matter to us as long as every routine is forced to produce the same number of extra lines.

Note that if we use a horizontal line to separate a pair of black and white points corresponding to a 1 in column  $c$  of  $M$ , we correspond it to choosing a row that covers  $c$ . The position of the horizontal line determines which row got chosen, as it is clear from the figure that all the pairs of black and white points corresponding to the 1s in a particular row  $r$  of  $M$ , will get separated by the same horizontal line due to the method of construction of the instance. But, we have to keep in mind that any algorithm that finds out the required set of axis-parallel lines of minimum cardinality, is going to pick up lines both in the horizontal as well as the vertical direction. If for every instance it can find out the solution by drawing only some horizontal lines, then there must be a problem in the algorithm as all instances of the *MILC* problem cannot be solved optimally by drawing lines along only one direction, like the

instance shown in Fig. 4.1 cannot be separated by 3 axis-parallel lines if they are restricted to one direction only. As in each column of  $M$ , there are two 1s, we have inserted 2 pairs of points in the instance corresponding to each column. However, there is a small trick that we followed in placing the two pairs of points corresponding to an edge  $e$  of the graph. Note that the two pairs are placed within the column corresponding to that edge  $e$  but slightly shifted from each other in the horizontal direction such that an imaginary dashed vertical line can be drawn to separate those two pairs as shown in the figure. A vertex cover for a graph ensures that at least one end of each edge is covered and hence the horizontal lines belonging to  $R$  will also ensure that one of the two pairs corresponding to each column is separated. To separate the rest of the pairs we will need some more lines possibly vertical. We add some more pairs of black and white points — one pair per column of  $M$  at the bottom of the figure. Note that each of these pairs are placed in such a way that the points belonging to the same pair have the same y-coordinate and they lie within the two dotted column separators that separate that particular column from its neighbors. This completes the reduction, which can be easily performed in polynomial time. A naive approach will take  $O(V \times E)$  time, whereas if the matrix shown in Figure 4.2(b) is not created explicitly, then it can be done in  $O(V + E)$  time.

**Claim 4.1** *Let the size of the minimum vertex cover for the graph  $G$  be  $n$ . In the figure constructed as above, any algorithm that will be able to separate all the pairs of black and white points using minimum number of axis-parallel lines must either do it by drawing  $|E|$  vertical lines and  $n$  horizontal lines, or otherwise give a solution from where a minimum vertex cover for the graph can be worked out.*

**Proof.** Note that the point pairs at the bottom of the figure will ensure that in the final solution at least one vertical line has to pass through each column.

In the figure the occurrence of solid vertical lines one per column is due to that fact. Again if this vertical line is aligned with the dashed line within the column, it separates only the pair of point at the bottom of the column, but if it is shifted to the the left or right by an appropriate amount, it can additionally separate one more pair of points. As we are looking for a solution that will separate all the multi-colored pairs using minimum number of axis-parallel lines, the vertical lines will rather separate one more pair than getting aligned with the dashed line in the column. Also note that any of these solid vertical lines can never separate both the pairs occurring within the same column that correspond to the two end-points of an edge of the graph, because of the way they were placed within the column. From the above discussion it follows that it is possible to separate all the pairs of points inserted, into monochromatic zones, by drawing  $n$  horizontal lines and  $|E|$  vertical lines, other than the  $|V| + 1$  row separators and  $|E| + 1$  column separators. Note that  $n$  horizontal lines placed in appropriate rows corresponding to the vertices occurring in the minimum vertex cover set will segregate at least one pair of points inserted per column. In each column, the other pair that is not separated (if any) can be separated simultaneously with the bottom pair, if the vertical line is placed intelligently by the algorithm.

We now argue that there cannot exist any solution with less number of lines separating all the pairs. Let us consider a solution set where the number of horizontal lines is  $n - k < n$ . Note that if we consider the graph  $G$ , then the vertices corresponding to these  $n - k$  lines will fail to cover at least  $k$  edges, otherwise all the vertices in the minimum vertex cover set is not essential and we face a contradiction. That means, at least  $k$  columns will have two pairs of points that are not separated, after we draw only the horizontal lines. Now the essential set of  $|E|$  vertical lines will at most be able to take care of one extra pair per column other than the bottom pair. Hence to separate the remaining

pairs we will need at least  $k$  more vertical lines. We already explained that a solution with less than  $|E|$  vertical lines is not possible to exist. Hence any solution set must contain at least  $|E| + n$  lines.

The last question that needs to be answered is, if there exists a solution set of size  $|E| + n$  separating all the pairs, but the number of horizontal lines is less than  $n$ , how can we identify the minimum vertex cover from that set. The answer lies in the discussion already done above. If such a solution has  $n - r$  horizontal lines, then there will be 2 vertical lines per column in  $r$  out of  $|E|$  columns. We have to choose any one end-point from each of the edges corresponding to those  $r$  columns and combine that with the  $n - r$  vertices corresponding to the horizontal lines to get a minimum vertex cover solution. Hence the proof.  $\square$

The above claim establishes that the *MILC* problem is NP-hard.

It is very easy to show that the *MILC* problem is in NP. Given  $n$  points in all, we have already shown that the axis-parallel lines that will constitute the final solution can be chosen from an universe of  $n - 1$  vertical lines and  $n - 1$  horizontal lines. The position of these  $2(n - 1)$  lines can be determined in linear time. Hence the work that remains is to make a yes-no choice for each line whether it will belong to the final solution or not, which can also be done in linear time by a non-deterministic algorithm. Hence we come to the following result.

**Result 4.1** *The decision version of the MILC problem is NP-complete.*

### 4.3.4 A fast heuristic

In this section, we present a greedy heuristic for the *MILC* problem that runs about 10 times faster than an alternative LP-based solution, whereas the cost

of the solution remains comparable. Gaur et al. [40] gave an LP-based 2-approximation algorithm for finding the minimum number of axis-parallel lines to stab a set of axis-parallel rectangles on the plane. Let this problem be named as *RSTAB*. *RSTAB* is also NP-complete [47].

**Lemma 4.1** *The MILC problem easily reduces to RSTAB problem in polynomial time.*

**Proof.** Given two sets  $B$  and  $W$  of  $p$  black points and  $q$  white points respectively on the plane. Consider the  $p \times q$  rectangles, each of which is defined by one member from  $B$  and one member from  $W$  at the end of one of its diagonals. It easily follows that the minimum number of axis-parallel lines required to stab the above set of rectangles is same as the minimum number of axis-parallel lines required to separate the points in  $B$  and  $W$  into monochromatic cells and vice versa. □

Thus, the LP-based 2-approximation algorithm proposed by Gaur et al. [40] can produce a solution of the *MILC* problem with same approximation ratio. As the linear programs that need to be solved for this purpose have 0-1 coefficient matrices, their time-complexity will be polynomial in the numbers of variables and constraints [122]. Finally, the time-complexity of the 2-approximation algorithm turns out to be  $O(r^5)$  for  $r$  rectangles [40]. Note that the number of rectangles  $r$  may be  $O(n^2)$  in the worst-case, for a total of  $n$  points of two different colors. We propose a much faster heuristic algorithm for the *MILC* problem that runs in overall  $O(n^3)$  time, and produces a solution of nearly same quality as that of the LP-based algorithm.

Our greedy heuristic is very simple. We will consider all the rectangles defined by the pair of points  $(p_b, p_w)$ , where  $p_b \in B$  and  $p_w \in W$ . Notice that, we may consider only the empty rectangles (not containing any member of

## Reducing Intersections for Rectilinear Steiner Trees

---

$B \cup W$ ) among them because, if all these rectangles are stabbed by the chosen set of axis-parallel lines, the other rectangles will be automatically stabbed. Considering the reduced set of rectangles, find out the line that stabs the maximum number of empty rectangles. Include that line in the solution set and then remove all those rectangles which are already hit. Then find the next line that hits the maximum number of rectangles in the remaining set. However, we choose the lines alternatively from the horizontal and the vertical set as it was experimentally found to give better results. Continue this process until all the rectangles are stabbed.

In the light of the above discussion we now list down the detailed steps of our heuristic  $H_f$  and also show a typical run in Figure 4.4.

- **Input:** A set  $B$  of  $p$  black points and a set  $W$  of  $q$  white points on the plane such that  $p + q = n$ .
- **Output:** A set  $L$  of horizontal and vertical lines separating the points into monochromatic cells.
- Step 1: Consider the  $p \times q$  rectangles each of which is defined by one black point and one white point from the Input set.
- Step 2: Form the reduced set  $R$  of rectangles by removing all the rectangles that has at least one point of  $B \cup W$ .
- Step 3: Sort the  $n$  points of  $B \cup W$  with respect to their x-coordinates. For each consecutive pair of points in the above sorted order draw a vertical line that separates them. Let  $V$  denote the set of these  $n - 1$  vertical lines.

Similarly, sort the  $n$  points in terms of their y-coordinates, and for each consecutive pair of points in the above sorted order draw a horizontal

line separating them. Let  $H$  denote the set of these  $n - 1$  horizontal lines.

- Step 4: For each of the lines in  $H$  and  $V$ , calculate the hit-number, i.e., the number of rectangles from  $R$  that it hits. Next, sort the lines of  $H \cup V$  in terms of their hit-numbers.
- Step 5: Select the line  $\ell$  with the maximum hit-number and put it into the output list  $L$ . Remove the set of all rectangles in  $R$  that are hit by line  $\ell$ .
- Step 6: Recalculate the hit-numbers of all the remaining lines.
- Step 7: If any rectangle is still left in  $R$ , GOTO Step 5. If the last line selected for  $L$  was horizontal (vertical), then this time select the winner only from vertical (horizontal) lines.
- Step 8: (Sweeping step) Consider the lines in  $L$  in the reverse order of entrance into  $L$ . If any line  $i$  is found to be redundant (i.e., all the rectangles in  $RSTAB$  problem are also hit by some line that entered into  $L$  after  $i$ ), then remove it from  $L$ .
- Step 9: Finally output the set  $L$  as the required set of lines separating the point-sets into monochromatic zones.

It is not difficult to show that the above heuristic runs in  $O(n^3)$  time. The time complexity will be dominated by Step 4, as it is equivalent to filling up a 0-1 matrix with  $O(n^2)$  rows (number of rectangles in the worst-case) and  $O(n)$  columns (number of lines in  $H \cup V$ ). Hence Step 4 may take  $O(n^3)$  time in the worst-case. The only other critical step is Step 6 as it may be repeated  $O(n)$  times in the worst-case. However Step 6 can take at most  $O(n^2)$  time, only when a line getting removed has a very high hit-number (say  $O(n^2)$ ).

# Reducing Intersections for Rectilinear Steiner Trees

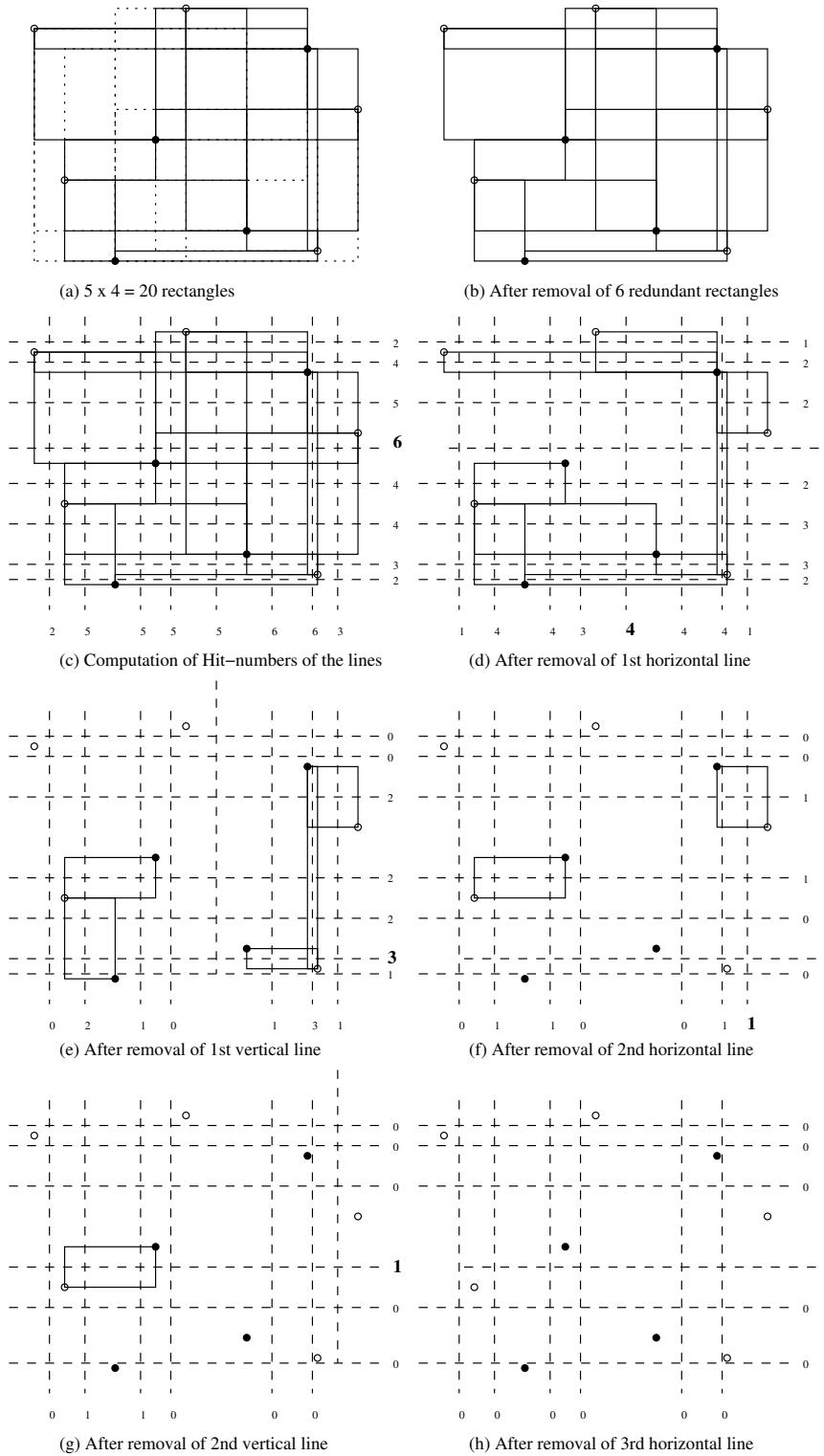


Figure 4.4: Sample run of our greedy heuristic

## Reducing Intersections for Rectilinear Steiner Trees

---

For solving the case of multi-color problem, we define the rectangles taking any two colors at a time. So for  $r$  colors, we need to consider  $r(r-1)/2$  set of rectangles, however a rectangle will be considered non-empty if a point of any color lies within it, and hence the final number of empty rectangles may be far less than  $n_1 \times n_2 \times n_3 \dots n_r$ , where  $n_1, n_2, \dots, n_r$  are the number of points of respective colors.

Table 4.1: Separation of bi-color points

Exp. #	# black points	# white points	Line Count		Time (sec)	
			Heuristic	LP-based	Heuristic	LP-based
1	5	5	2	2	0	0.625
2	10	10	8	7	0.015	0.547
3	20	20	12	11	0.015	0.641
4	40	40	15	16	0	0.532
5	60	60	25	22	0.156	0.890
6	90	90	34	33	0.093	2.235
7	100	1	2	2	0.016	0.109
8	100	100	39	37	0.141	3.047
9	100	150	43	42	0.219	5.531
10	120	120	43	43	0.219	6.343
11	150	140	49	43	0.312	9.469
12	150	150	46	45	0.390	10.703
13	175	175	53	51	0.657	23.921
14	200	200	54	52	0.703	25.438
15	200	100	46	46	0.313	6.968
16	250	200	62	62	6.047	57.047

### 4.3.5 Performance of our greedy heuristic

We have implemented the greedy heuristic for the *MILC* problem on a pentium-based machine with 256 Mb RAM. The program is written in the C language and used graphics libraries to draw the solutions of moderate size. We also implemented the LP-based solution and used an LP package named `lp_solve` Release 2.0 available in the public domain to solve the main LP problem. The other two similar LP problems that were avoidable have been replaced by a linear-time routine, so that the comparison with our heuristic becomes more realistic.

Table 4.1 gives the results of our experiment for points of two colors whereas Table 4.2 is for multi-colored points. The cost of the solutions of the two techniques are comparable though the results of the LP-based technique being better on an average. However the speed of the LP-based technique is a serious hindrance in using it for problems beyond a certain size and our heuristic is at least about 10x faster in almost all cases.

Table 4.2: Separation of multi-color points

Exp. #	# colors	Total # points	Line Count		Time (sec)	
			Heuristic	LP-based	Heuristic	LP-based
1	6	83	24	25	0.109	0.345
2	8	125	35	33	0.063	1.765
3	5	119	33	33	0.328	0.532
4	4	249	51	47	1.016	15.670
5	8	209	49	46	0.469	11.249
6	7	165	42	43	0.172	4.968
7	8	261	57	54	3.281	26.686
8	6	252	55	51	1.469	18.405

## 4.4 Solution methodology for the intersection problem

We will now utilize the fast heuristic  $H_f$  for monochromatic partitioning described above to reduce the crossing number of multi-color Steiner trees.

Let us consider  $k$  nets  $\{n_1, n_2 \dots n_k\}$ , and their corresponding sets of terminals  $\{T_1, T_2 \dots T_k\}$ . We note down the positions of the terminals on the plane and mark the point-positions corresponding to terminals of  $T_i$  using color  $c_i$ . Let  $P_c$  be the set containing all the colored points corresponding to the point-positions mentioned above. Note that the set  $P_c$  is well-defined as all these points are distinct as they correspond to distinct terminals.

We now want to connect all the points of the same color using rectilinear lines keeping two objectives in mind. First the total length of line segments required to interconnect the points of the same color should be as less as possible. We assume that the line segments used to connect the points with color  $c_i$  also have the same color. The second objective is to keep the total number of crossings between segments of different colors as low as possible. If we consider the first objective only, the problem remains computationally hard. Also it is possible to cite many such instances so that all the feasible solutions are independent of the second objective, i.e., the optimal solution is the one that optimizes the first objective only. Consider an example, where the colored trees do not intersect at all. This is of the above type. So, in this case the optimization problem with dual objectives will have to optimize the first parameter only. Hence the problem with dual objectives is also computationally hard.

In this discussion, we directly concentrate on giving a plan for a heuristic that tries to satisfy both the objectives in a reasonable way.

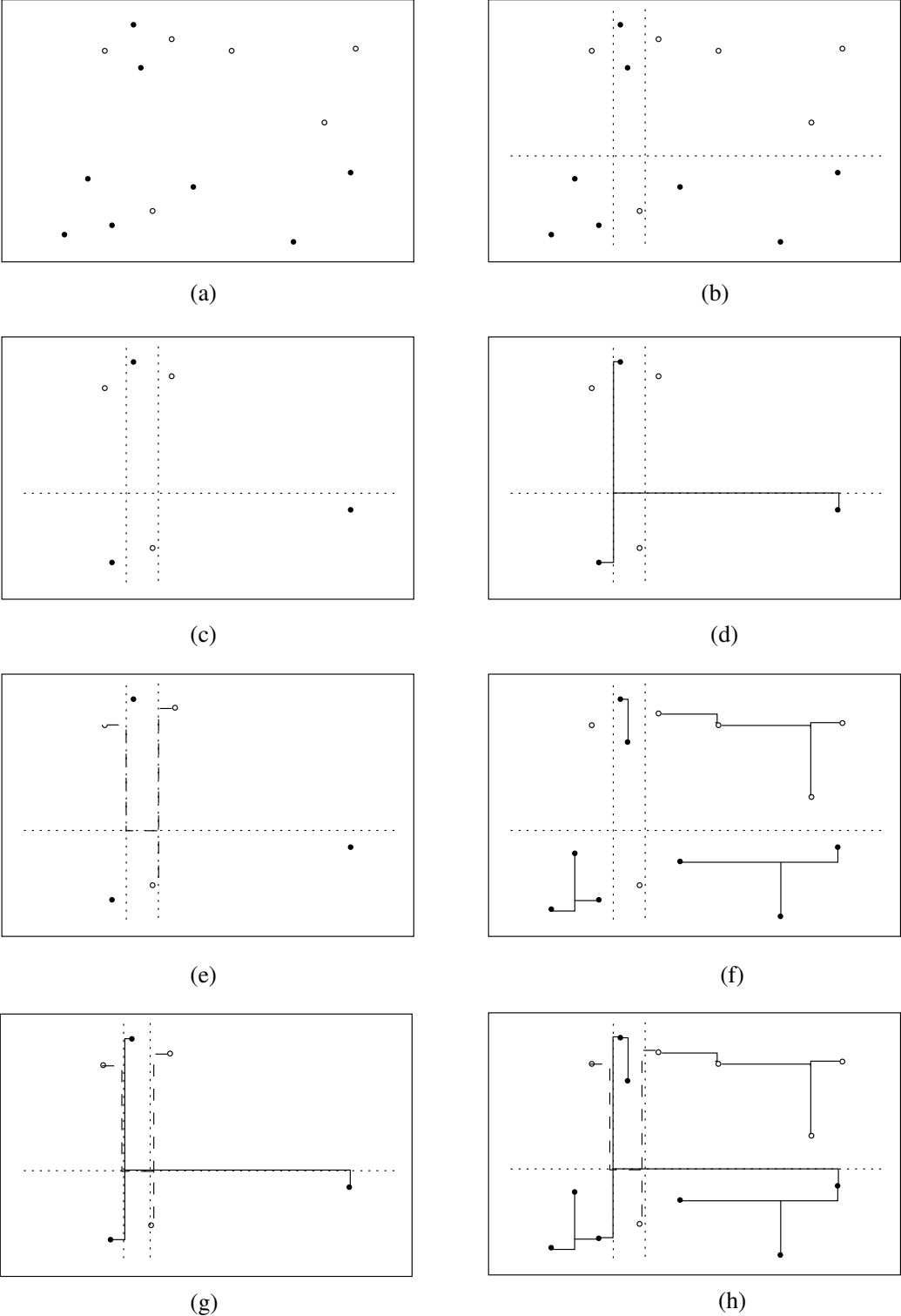


Figure 4.5: A typical example showing the steps of the heuristic  $H_s$

### 4.4.1 Working principle of the heuristic $H_s$

We now describe the different steps followed by our heuristic  $H_s$  and in Figure 4.5 we show how it solves the problem in case of a small but typical example.

We first run the heuristic  $H_f$  on the set of colored-points  $P_c$  so that the points get segregated into monochromatic cells by a small set of axis-parallel lines  $L$  (Figure 4.5(b)). There may be some cells that will contain more than one point. From each of those cells, we choose a representative point to get a reduced set of points  $P_r$  on the plane (Figure 4.5(c)). For each distinct color  $c_i$ , we then generate an RST  $t_i$  to connect all the points of  $P_r$  that have color  $c_i$  (Figure 4.5(d) and (e)). We call them the global RSTs and let the set of these RSTs be  $R_G$ . However, while generating these global RSTs we put an additional restriction as described below. Suppose an edge  $e \in t_i$  will connect two points  $p$  and  $q$  of color  $c_i$ . Starting from  $p$  it either takes a vertical route or a horizontal route. Once it hits the boundary of the monochromatic cell to which it belongs, it starts using only those isothetic line segments that are subsets of the lines belonging to  $L$ . It may use as many line segments as required guided by the strategy of the heuristic  $H_s$ , till it hits the boundary of the monochromatic cell in which  $q$  is placed. It then uses a final horizontal or vertical segment to get connected to  $q$ . Once these global RSTs are generated, we go back to each of the monochromatic cells and locally generate an RST to connect all the points of the same color present within that cell (Figure 4.5(f)). The RSTs thus generated are called local RSTs and let the set of such RSTs be  $R_L$ . We call two RSTs  $X$  and  $Y$  to have intersected, if any edge belonging to  $X$  geometrically crosses any edge belonging to  $Y$ . The following fact easily follows from the method of construction.

**Remark 4.1** *Any two local RSTs do not intersect among each other. A local*

*RST of any color will not intersect with a global RST of different color. Two global RSTs however may have intersections among themselves (Figure 4.5(g)).*

This method will actually increase the total wire-length required to complete the full routing. If we had directly used our heuristic to connect the points of same color forming an RST instead of doing the monochromatic partitioning, the total wire-length requirement would have been definitely less. However our method will ensure that the number of crossings amongst multi-color edges will be less compared to the direct approach. This may in turn lead to save the expensive vias, as each crossing will require one of the wires to change layer using vias. In the next section we present some experimental results to show the effectiveness of our heuristic in this regard.

### 4.5 Experimental results

We have tested the performance of our heuristic on a set of randomly generated problem instances of various sizes. Table 4.3 and Table 4.4 present the comparative study of our strategy against the direct approach in terms of total wire-length and number of crossings. The experimental results have two broad divisions. One of them is called RST-based, which is exactly same as the discussion given in the previous section. The other one is called MST-based, where the local RSTs are being replaced by MSTs. Note that these local MSTs are not rectangular but drawn in the Euclidian metric. In case of RST-based experiment, we compare our results with big RSTs being drawn directly with the original set of points, whereas in case of MST-based experiment we compare our results with big MSTs being drawn directly with the original set of points. As our heuristic-based approach generates the trees in two steps, one at global level and the other at the local level, we have named our approach as hybrid approach.

## Reducing Intersections for Rectilinear Steiner Trees

Table 4.3: Performance of heuristic for two colors

Type of Exp.	Number of terminals (color 1)	Number of terminals (color 2)	Total wire length			Number of Crossings		
			Direct approach (100 %)	Hybrid approach	% more in hybrid	Direct approach	Hybrid approach (100) %	% more in direct
MST based	10	20	4653	7328	57	2	2	0
	10	50	5399	7894	46	7	4	75
	20	25	5493	8529	55	8	6	33
	40	60	8668	11549	33	13	10	30
	50	80	9493	13158	38	22	18	22
	80	100	11565	15812	36	27	23	17
	100	110	12540	17272	37	33	29	13
	40	120	10329	14209	37	22	16	37
	20	200	11301	17126	51	24	16	50
Average					43.3			30.8
RST based	10	20	5521	7680	39	3	2	50
	10	50	6822	8584	25	8	4	100
	20	25	7000	8828	26	7	6	16
	40	60	10513	11872	12	18	10	80
	50	80	11525	13654	18	26	18	44
	80	100	14104	16265	15	32	23	39
	100	110	15374	17811	15	36	29	24
	40	120	12623	14782	17	20	16	25
	20	200	14061	18340	42	26	16	62
Average					23.2			48.9

Note that, with some increase in wire-length, heuristic  $H_s$  has been significantly successful in reducing the number of crossings among the edges of different colors in both types of experiments. Consider Table 4.3. Except for a few very small examples our approach has always produced far less number of crossings. Also note that, the number of crossings remain same in either type of experiment in case of hybrid approach. This is because, the remark made in the last section about RST-based experiment holds good for MST-based experiment also. Any local MST will not intersect with any other local MST, also it will not intersect with the global RST. This again follows from the method of construction, as a local MST is always bounded within its cell. The only intersections that are reported come from the edge crossings of global RSTs, which are same for the both the types. There is a percentage of increase in the total wire-length from the direct approach as predicted earlier. However, the percentage is much less in case of RST-based experiment, which is in fact more realistic as routings in VLSI domain are mostly manhattan. The percentage

## Reducing Intersections for Rectilinear Steiner Trees

increase in total wire-length has been high for the MST-based experiment, because in the hybrid approach the global tree is rectilinear which obviously takes more length than its Euclidian counterpart. These experiments have been carried out in an ordinary pentium-based machine. The CPU time required has been negligible and also comparable for both the approaches, so for brevity we did not report them. The results obtained for multi-color cases as

Table 4.4: Performance of heuristic for more than two colors

Type of Exp.	Number of nets (colors)	Number of terminals (points)	Total wire length			Number of Crossings		
			Direct approach (100 %)	Hybrid approach	% more in hybrid	Direct approach	Hybrid approach (100) %	% more in direct
MST based	3	45	7271	9785	34	19	12	58
	3	35	4913	6596	34	7	4	75
	3	90	10007	12783	27	31	26	19
	4	80	10910	13963	27	33	27	22
	5	50	10000	12852	25	38	26	46
	5	60	9783	12167	24	33	24	37
	5	79	12847	16212	26	53	47	12
	6	51	10176	12699	24	37	32	15
	7	86	14653	19590	33	92	71	29
	7	69	13940	16991	21	77	63	22
8	84	14169	17700	24	79	63	25	
Average					27.2			32.7
RST based	3	45	9018	10013	11	20	12	66
	3	35	5907	6947	17	11	4	175
	3	90	12176	13114	7	33	26	26
	4	80	12992	14095	8	37	27	37
	5	50	11948	12936	8	44	26	69
	5	60	11560	12318	6	38	24	58
	5	79	15436	16298	5	62	47	31
	6	51	11889	12776	7	39	32	21
	7	86	17946	19632	9	91	71	28
	7	69	16185	17020	5	75	63	19
8	84	17111	17774	3	85	63	34	
Average					7.8			51.3

shown in Table 4.4 are in the similar note. However the savings in the number of crossings is more conspicuous from this table. Here, for the RST-based experiment, the average percentage of increase in wire-length is only about 8, whereas the average percentage of reduction in the number of crossings is much more. In a practical situation, the number of multi-terminal nets will be much more than two and hence our proposed method will also be much more effective in such cases. The two facts that the results get better with increas-

ing number of colors and also in the RST mode of the experiment makes our approach quite useful in the practical scenario.

### 4.6 Conclusion

In this chapter, we have presented a novel heuristic to reduce the crossing number of multi-color rectilinear Steiner trees. This may in turn lead to reduce the number of vias required for global routing in a VLSI chip and hence better utilization of the metal layers for detailed routing. We have first proved that separating a set of multi-colored points on the plane into monochromatic cells using axis-parallel lines is computationally hard. We have also proposed a fast heuristic to solve this problem and compared our results with that of an existing LP-based technique that can be adopted to solve the same problem. Our heuristic runs about 10 times faster than this technique whereas the cost of the solution remains comparable. Finally, we have used this heuristic for reducing the number of intersections among multiple rectilinear Steiner trees, where the trees for distinct nets are marked with different colors. The experimental results reported show the effectiveness of our approach.

# Chapter 5

## Geometric Analysis of Hot Spots

### 5.1 Introduction

There has been a dramatic increase in the power consumption of ULSI circuits in recent times. This has led to a high operating temperature thereby affecting the timing concerns quite seriously. Even under normal operations the temperature can rise by several tens of degrees above the ambient temperature. According to ITRS [135], power density (energy dissipated per unit area per unit time) in a chip is rising significantly because of higher clock-rate and increasing device density. In fact, by 2010 power density in some regions of the chip may rise to  $200W/cm^2$  for 45nm technology. The maximum sustained power over a specified limit may cause excessive heating of the device ultimately leading to its failure. Design and placement of heat sinks as well as thermal vias have become issues of utmost importance. Other than temperature rise, a high on-chip thermal gradient can affect both performance and reliability significantly. Some researchers have addressed the problem of identification of hot spots

in VLSI chips that mostly involve complicated heat diffusion equations [24], whereas others have proposed an alternative placement scheme to cool down a hot chip [127, 25, 52, 51]. However, in this thesis we present an analysis technique that is based on simple computational geometry. Recently Kang has highlighted the need for new thermal designs in order to cope up with the challenging scenarios ensuing from the high-scale integration [56]. Miranda et al. had successfully used the finite volume method approach to investigate maximum temperature rising on a CPU motherboard [96]. CMOS fabrication is now being implemented with 90 nm technology, and soon will be moving to 65 nm, where power dissipation has become a major problem because of both active and leakage power loss. Design of several power saving techniques at the circuit and architectural level, and heat sink design thus have assumed great importance to industry [105].

## 5.2 Models for thermal analysis

We have proposed several models in this chapter, which may facilitate in identifying the hot spots/zones in a VLSI chip. First in the continuous domain we have used the concept of a unit circle model to calculate the local thermal effect at a point due to the heat being dissipated from several point heat sources distributed over the chip plane. We establish that a point on a chip can become very hot due to the conduction effects of other heat sources, although it may not have a heat source in its immediate vicinity. In our model, the heat loss due to radiation has been ignored. If it is to be considered, an appropriate heat loss function has to be incorporated.

Next we have studied the same issue in a grid-based model to exploit the advantages of discrete domain computation. Here also we have noted similar results. Further we have proposed the concept of a sweep window in order

to identify hot zones instead of singularities. Efficient algorithms exist in the literature to identify such a window [99]. All the models proposed in this chapter establish one important finding that a non-source point can become hotter than some of the source points due to the relative positioning of the sources and identifying them is critically important to ensure the thermal safety of the chip.

### 5.3 Time-invariant heat sources

The first study is made with the assumption that there are constantly active (i.e., always *on*) heat-generating sources placed randomly throughout the chip. For continuous thermal sources, we use the terms heat and power interchangeably without loss of generality. We also assume that heat from the sources is being propagated through the 2D surface of the chip floor without being dissipated in the ambience. The objective is to identify the zones on the floor, which have heat content greater than a certain threshold. Based on the permissible locations of the sources and the observation points, we categorize this problem into continuous and discrete domains.

#### 5.3.1 Continuous spatial domain

The position of a heat source may be any point on the chip floor which is assumed to be a 2D plane. In the unit circle model, the contribution of a point heat source  $S$  at any target point  $T$  is expressed as the amount of heat from  $S$  received within the unit circle centered at the point  $T$ . This unit is same as that of the distance between  $S$  and  $T$ , and may be related to the minimum layout dimension of the chip. The cumulative heat received at the point  $T$  is evaluated as the linear superposition of the amounts received at  $T$

from all heat-generating sources on the chip.

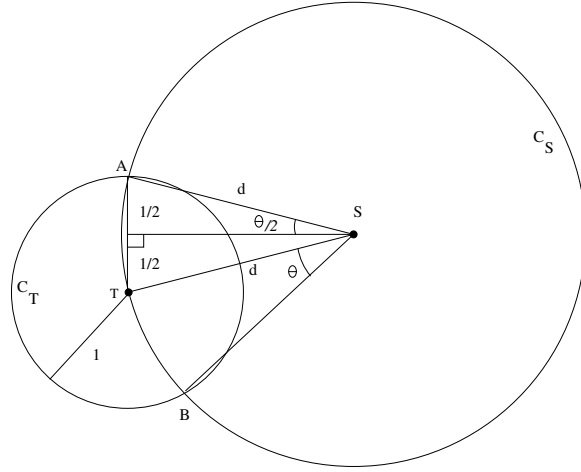


Figure 5.1: Unit circle model of heat received at point  $T$

As illustrated with Figure 5.1, let a heat source at a point  $S$  generate an amount  $Q$ , henceforth denoted as the strength of the source  $S$ . Let the target point  $T$  be at a Euclidian distance  $d$  from  $S$ . Consider the circle  $C_T$  of unit radius with center  $T$ , and the circle  $C_S$  of radius  $d$  centered at  $S$ . Let  $C_T$  and  $C_S$  intersect at the two points  $A$  and  $B$ . Let the angle subtended by the arc  $ATB$  of  $C_S$  at the center  $S$  be  $2\theta$ . The contribution of heat from  $S$  at  $T$  is  $Q\theta/\pi$  (i.e.,  $Q * \frac{2\theta d}{2\pi d}$ ). Using simple geometry we can derive that  $\sin(\theta/2) = \frac{1}{2d}$ , or  $\theta = 2 \sin^{-1} \frac{1}{2d}$ . Hence, the heat contribution of source  $S$  at  $T$  is given by  $2Q(\sin^{-1} \frac{1}{2d})/\pi$ .

Our concerns are the hottest points on the chip. Intuitively, the source points definitely belong to the above class. But the more pertinent question is whether these are the only points that need to be considered. The question may be re-phrased as follows: does there exist any non-source point on the floor with heat content greater than that of any of the source points?

Our observations, reported in this chapter, answer in the affirmative. Before we proceed further, we point out two special cases of the unit circle model based on the distance  $d$  between  $S$  and  $T$  : (i)  $0.5 < d < 1$ , and (ii)  $0 < d \leq 0.5$

## Geometric Analysis of Hot Spots

---

(vide Figure 5.2). In the boundary case when  $S$  lies on  $C_T$ ,  $\theta$  is equal to  $\pi/3$  as  $SAT$  becomes an equilateral triangle. So in case (i), the angle  $2\theta$ , as defined earlier, is greater than  $2\pi/3$  and consequently more than  $1/3$  of the heat emanating from  $S$  reaches the unit circle centered at  $T$ . In case (ii),  $T$  is nearer to  $S$  and hence the circle with radius  $d$  around  $S$  now lies entirely within the unit circle at  $T$ . Hence the unit circle  $C_T$  receives the entire heat of  $S$  in this case.

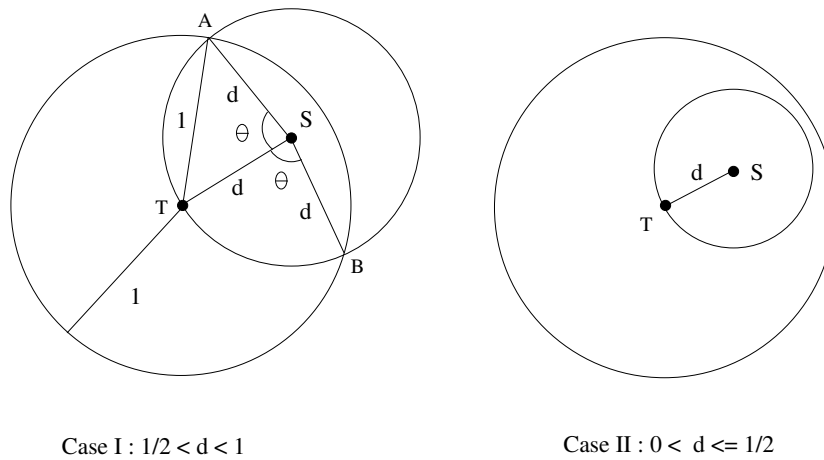


Figure 5.2: Special cases of the unit circle model

### Illustration of thermal victim

As shown in Figure 5.3, let us consider a regular polygon with 8 sides, where the radius  $r$ , i.e., the distance from the center of the polygon to each vertex is 2. An active source of unit strength is placed at each of the 8 vertices. Using the formula derived above, we plotted the cumulative heat received at each point along the radial line joining the center and one of the vertices. Because of symmetry the case is identical along all the 8 radial lines. In the plot the cumulative power attains a maxima at a point with distance tending towards 1.5 from the center along the radius. The cumulative power at any of the vertices is  $P = 1.90252$ , where the self-contribution from the active source

is 1 unit and the net contribution from the other 7 sources is 0.90252 units. Along the radius the cumulative power at a point  $R$  whose distance is 1.495 from the center is found to be 1.91534, which is greater than the cumulative power at any of the vertices. Around  $R$ , there are other points also where the cumulative power crosses the threshold  $P$ . The power patterns changed to some extent when some of the sources are removed, i.e., when 6 or 7 of the 8 sources are present. We get similar results even for a square.

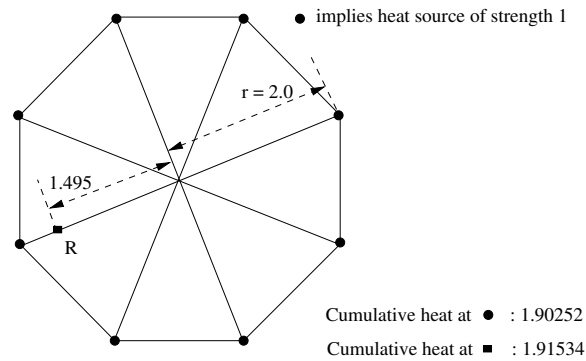


Figure 5.3: Thermal pattern within an octagon

In Figure 5.4, we consider 10 heat sources being placed randomly on a chip floor, their strengths varying from one unit to three units. Again using the previous formula the value of the cumulative heat is evaluated at 400 (20 x 20) probe points on the floor uniformly distributed over a set of horizontal and vertical lines, each being located at a distance of 0.5 units from its adjacent one. The threshold value is defined as the minimum of the cumulative power at any of the active source points. For clarity, we have plotted the cumulative heat only for those probe points where the value exceeded the threshold and there are 41 non-source points belonging to that group. A few of them as shown in the figure are located away from the sources. In Section 5 on Experimental Results, we have reported several cases with active heat sources dissipating varied amount of power, where the cumulative power at a non-source point crosses the threshold. Hence in order to prevent the chip from overheating, we

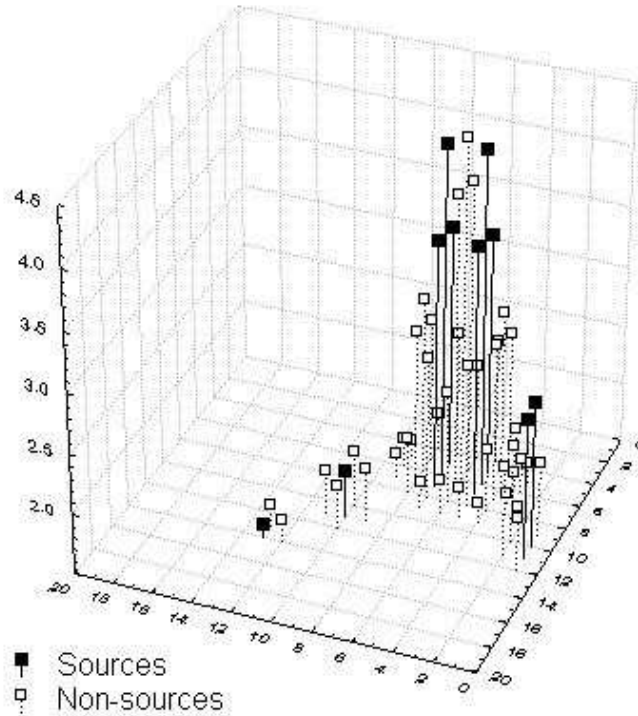


Figure 5.4: Typical thermal distribution for 10 sources

may also have to guard certain points or zones in the floor that do not contain any active source.

### 5.3.2 Discrete spatial domain

In order to simplify the computation in the continuous domain, we have considered a grid-based model in the discrete domain. We have taken a uniform grid of suitable resolution and assumed that the source points are placed only at a subset of the grid points. We are interested in evaluating the cumulative power only at the grid points. The propagation of heat energy assumed in this model is as discussed below.

Consider the uniform grid shown in Figure 5.5 with grid resolution (i.e., distance between two adjacent grid points) of 1 unit. There is only one active

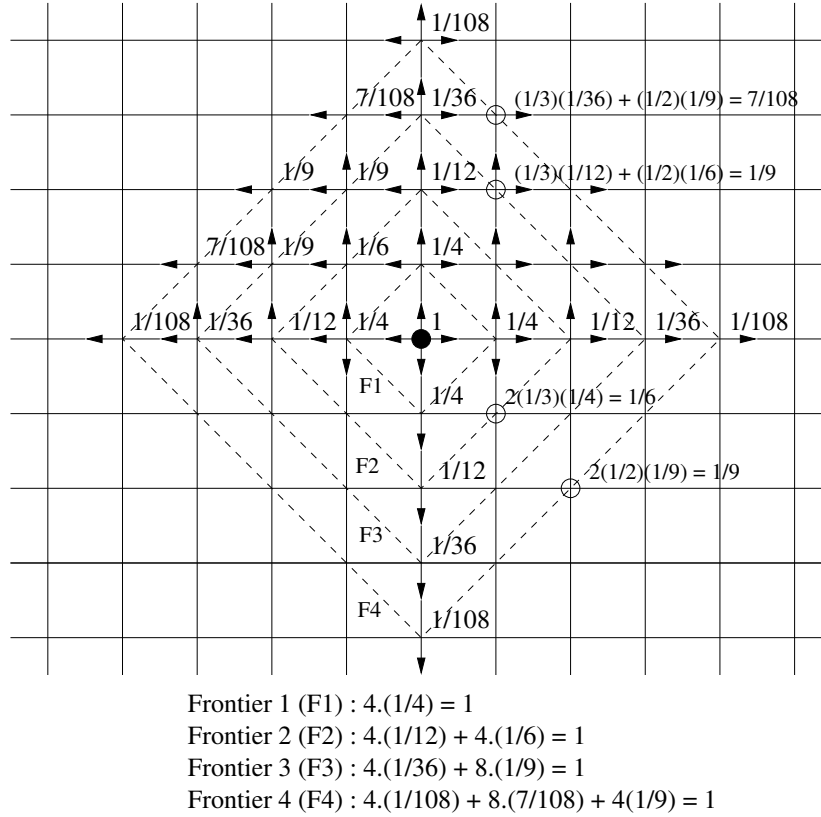


Figure 5.5: Propagation model for discrete domain

heat source of unit strength present on the floor, shown as a solid circle. In this diagram we have shown, the value of the cumulative power at the other grid points due to the presence of this single heat source. A frontier  $k$  consists of the grid points that are at a *manhattan* distance of  $k$  from the source. A naive model assumes that the unit power gets equally distributed in the ratio  $1/(4k)$  at each of the  $4k$  points present on the  $k^{th}$  frontier.

However we have considered a more accurate model. Each of the four nearest neighbors of the source receive  $1/4$  units of heat. From thereon each of the grid points that lies on the two grid lines containing the source, transmits  $1/3rd$  of its received power to its three neighboring grid points away from the source. All other points on the grid transmit half of their received power to two of their neighboring grid points in a direction away from the source.

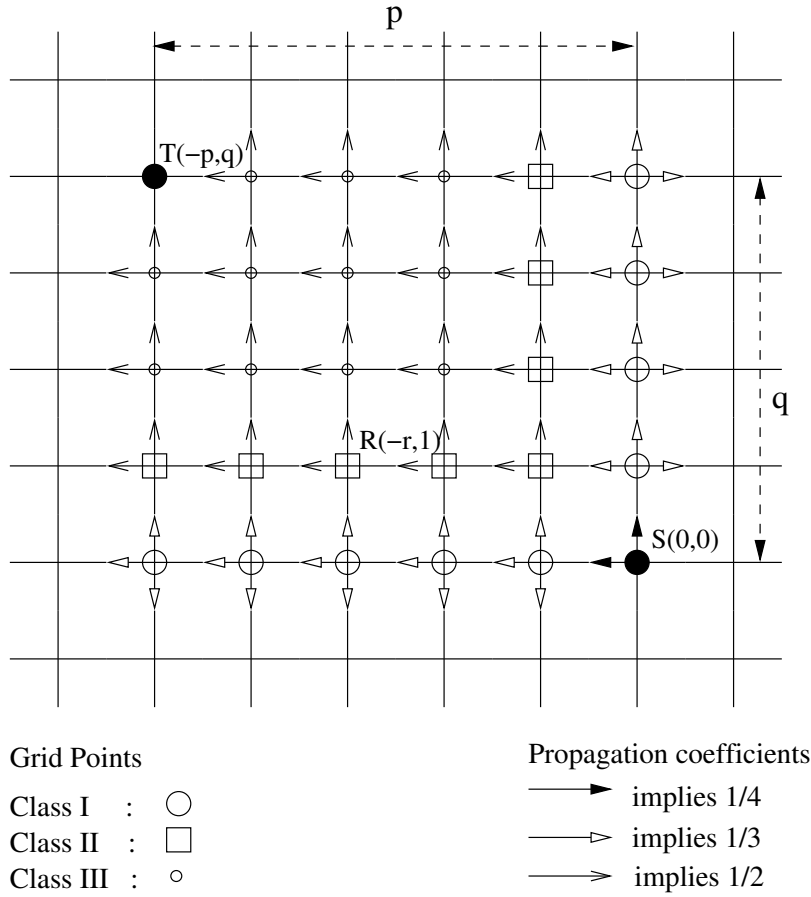


Figure 5.6: Propagation coefficients in the grid

The direction of heat flow and propagation coefficient along each edge of the grid is shown in Figure 5.6 for one quadrant. Reflection symmetry holds for the other three quadrants. If however a boundary point or a corner point is reached, the power retransmission stops there immediately. The sumtotal heat propagating out of each frontier remains same which is equal to unity by the law of conservation of energy.

Given the grid-coordinates of the active heat sources, we can take a simulation based approach to calculate the cumulative power at all the grid points using the above propagation rule shown in the Figure 5.6. But to evaluate the cumulative power at a given target point  $T$  on the grid, we have derived a closed form expression, thereby rendering the computation to be more ef-

ficient. Let the horizontal and vertical distances between  $T$  and  $S$  along the two axes be  $p$  and  $q$  respectively. Without loss of generality we assume that  $S$  is placed at the point  $(0, 0)$ . Note that  $p$  and  $q$  are magnitudes only, due to symmetry their signs do not affect the formulae.

Consider Figure 5.6 again. There are three classes of grid points. The class I points, lie on the same horizontal(vertical) grid line as  $S$ , i.e.,  $q(p) = 0$ , and the cumulative power at such a point is given by  $1/(4 * 3^{p-1})$  ( $1/(4 * 3^{q-1})$ ).

The class II points are those having a distance from the source  $S$  along one of the axes as unity. The power  $P$  at the point  $(p, 1)$  for  $p > 1$ , is given by the recurrence relation

$$P(p, 1) = P(p, 0) * (1/3) + P(p - 1, 1) * (1/2).$$

Substituting the values of  $P(p, 0)$  from above and

$$P(1, 1) = 2 * (1/4) * (1/3), \text{ we get the solution}$$

$$P(p, 1) = (1/(3 * 2^{p-1})) - (1/(2 * 3^p)) \text{ and similarly}$$

$$P(1, q) = (1/(3 * 2^{q-1})) - (1/(2 * 3^q)).$$

For class III points, we have the following recurrence  $P(p, q) = (1/2)*P(p-1, q) + (1/2)*P(p, q-1)$ ,  $p, q > 1$ . We can now formulate the cumulative power at any of the class III points (say  $y$ ) in terms of the power values of those class II points that contribute towards  $y$ . In Figure 5.6, the class II points contributing power to  $T$  have coordinates  $(-x, 1)$ ,  $1 \leq x \leq p$  and  $(-1, y)$ ,  $2 \leq y \leq q$ .

Now what is the contribution of any of the above class II points towards  $T$ ? Let the coordinate of such a point  $R$  be  $(-r, 1)$ . Power can reach through any of the monotone paths from  $R$  to  $T$  along the grid. The propagation coefficient along any edge on these paths is  $1/2$ . Also the length of each of these paths is equal to the manhattan distance between  $R$  and  $T$ , which is  $(p - r) + (q - 1)$ . The number of such paths from  $R$  to  $T$  is given by the respective Catalan number, which is  $\binom{p-r+q-1}{q-1}$ . Hence the contribution of  $R$  towards  $T$  is

## Geometric Analysis of Hot Spots

---

$[1/(3*2^{r-1}) - 1/(2*3^r)] * \binom{p-r+q-1}{q-1} * (1/2)^{p-r+q-1}$ . This should have given the total power by summing over all candidate class II points. However we need to consider the following. Some of the power that gets emanated from a class II point also contributes towards a neighboring class II point and that gets added twice. So for point  $R$  we have to consider only that fraction of its power, which it receives from a class I neighbor. If we now sum up over all the candidate class II points, the final expression for cumulative power at  $T$  is given by  $P(p, q) =$

$$\sum_{k=1}^p (1/(3 * 4 * 3^{k-1})) * \binom{p-k+q-1}{q-1} * (1/2)^{p-k+q-1} +$$

$$\sum_{k=1}^q (1/(3 * 4 * 3^{k-1})) * \binom{q-k+p-1}{p-1} * (1/2)^{q-k+p-1}$$

By symmetry,  $P(p, q) = P(-p, q) = P(p, -q) = P(-p, -q)$ .

These formulae are used for our experiments in the discrete domain where we have more than one steady active heat source placed at randomly chosen points of a uniform grid. The results depict that the cumulative heat at few of the passive grid points exceeds that at some of the grid points containing active heat sources, similar to continuous domain.

### 5.3.3 Window model in discrete domain

Our third study is the cumulative heat received within a rectangular window (zone) aligned with the grid. For a given position of the window, let the cumulative power at each grid point covered by the window be denoted by the *Window Power*. Instead of comparing the cumulative power at a single grid point, we now sweep a rectangular window of fixed size and aspect ratio and then compare the *Window Power* at different positions.

Consider Figure 5.7. In this model also we similarly observed that at some positions, where the window does not contain any active source, the *Window Power* actually records a higher value than some of the positions, where the

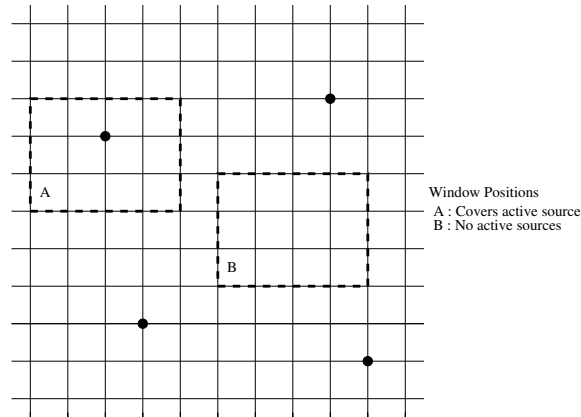


Figure 5.7: Moving window over the discrete grid

window actually contains an active source(s). If we define a threshold value for the *Window Power*, and there are several overlapping windows crossing that threshold, then we can conclude that we should thermally guard the entire zone obtained by taking a union of those windows.

## 5.4 Time-varying heat sources

Finally comes the most generalized model where the active source points may be time-varying. Depending upon the switching characteristics of the chip, for a given period, some of them may behave like a passive point also. Each active source has a duty cycle associated with it. Although a detailed study of this model is currently in progress, we have already observed a very interesting feature. In the previous models the cumulative power at any passive point achieving the maximum value on the chip floor was a rare phenomenon. However in this model, with active sources at random positions, we often encounter such counter-intuitive results, i.e., zones with no active sources turned out to be the hottest zone of the floor.

## 5.5 Experimental results

For the continuous domain, we used Mathematica 5.0 to simulate the effect of active sources placed at random points on the floor. Keeping the size of the floor same, we varied the number of sources from 5 to 50. We have studied five trial runs, keeping the number and the power strength of the sources fixed, just allowing the position of the sources to vary.

Table 5.1: Results for the continuous domain

# src pts	# probe pts	average $n_1$	average $n_2$
5	4500	68	0
10	14000	185	1
20	30500	826	15
40	56000	1673	35
50	60000	1871	46

We observed that the results did not depend much on the randomness of the position of the active sources. We have run this experiment as a biased Monte Carlo. As in the previous section we have observed that in the continuous domain the relatively hot points lie near the active sources, during simulation, we consider more of those points for evaluation of the cumulative power. We actually consider a fine grid around each source point and evaluate the cumulative power at each of those points along with the source points. Also across the whole floor we consider a relatively coarse grid and evaluate the power at all the grid points of this coarse grid. The formula derived from the unit circle model is being used for the calculation. In Table 5.1, we have reported the average values over the 5 trials, where  $n_1$  stands for the number of non-source points where cumulative power crossed the threshold, while  $n_2$  stands for the number of non-source points that lie outside the fine grid and still crossed the threshold. The threshold value is the minimum of the total

power at the active source points including contribution from other sources. We find that the number of points away from the sources but still crossing the threshold is comparable to the number of source points.

Table 5.2: Results for the discrete domain

# src pts	# probe pts	min. trial value of $n$	max. trial value of $n$
5	51 x 51	3	7
10	51 x 51	7	157
20	51 x 51	46	164
40	51 x 51	289	483
50	51 x 51	307	702

We have performed a similar experiment in the grid-based model implemented in C. We assume a finite number of sources distributed randomly over a 50 x 50 grid (i.e. 51 x 51 probe points), with the number of sources varying from 5 to 50. For each run, we have kept the number of sources and the power of each source fixed, but their positions are varied randomly over the 5 trials. Here also the threshold is set at the minimum of the cumulative power values at the active source points. The results are shown in Table 5.2, where  $n$  stands for the number of non-source points crossing the threshold. We have evaluated the power at all the points of the uniform grid. Here the percentage of non-source points crossing the threshold is found to be much higher compared to the earlier experiment.

For performing the experiment in the window based approach we again consider a finite number of sources distributed randomly over a 50 x 50 grid. Given a window-size we define the window-threshold  $P_{window}$  as a product of the minimum cumulative source power times the number of grid points covered by the window. The results of the experiment is shown in the Table 5.3. The number of sources is varied from 5 to 40. However, here we consider four trials

## Geometric Analysis of Hot Spots

---

Table 5.3: Results for window model

# src pts	size of window	Trial #	$n_1$	$n_2$
5	3 x 3	1	0	0
		2	0	0
	5 x 5	3	0	0
		4	0	0
10	3 x 3	1	26	0
		2	101	19
	5 x 5	3	26	0
		4	0	0
20	3 x 3	1	57	0
		2	82	2
	5 x 5	3	18	0
		4	11	0
40	3 x 3	1	136	12
		2	378	139
	5 x 5	3	187	20
		4	854	243

in each case, two for a 3 x 3 window and two for a 5 x 5 window. In the table  $n_1$  stands for the number of window positions for which the cumulative power crosses the threshold  $P_{window}$ , while  $n_2$  stands for the number of those windows that do not contain any heat source but still crosses the window-threshold.

Finally since the threshold power  $P_{th}$  affects the window-threshold  $P_{window}$ , we study the effect of varying  $P_{th}$  from the minimum cumulative source power to maximum initial source power. We consider the last trial from Table 5.3 with 40 heat sources, keeping their strengths and positions fixed, but varying  $P_{th}$  from 6.77 units to 17.92 units. As reported in Table 5.4, this caused

Table 5.4: Power window results with varying threshold

	Trial #	$P_{th}$	$P_{window}$	$n_1$	$n_2$
# src pts = 40	1	6.77	243.75	854	243
	2	9.00	324	299	37
	3	11.23	404.28	46	0
Window Size = 5 x 5	4	13.46	484.56	0	0
	5	15.69	564.84	0	0
	6	17.92	645.12	0	0

$P_{window}$  (36  $P_{th}$  from 6 x 6 probe points) to vary from 243.75 to 645.12 units. As expected  $n_1$  and  $n_2$  decrease as window-threshold increases.

## 5.6 A hybrid approach

We now propose a methodology to speed-up the simulation-based method taking the help of Voronoi Diagram, a well-known concept of computational geometry. Voronoi Diagrams are widely used in a vast range of areas like physics, astronomy, biology, robotics as well as social geography [11]. In the next section, we explain how multiplicatively weighted Voronoi diagrams can be used to perform a fast but approximate prediction of the hot zones, which then can help in reducing the simulation overload. Finally, an accurate prediction of the hot zones can be obtained by following a hybrid approach that first uses geometry and then simulation. The thermal model being used for the following analysis is the continuous domain *unit circle model* [88].

### 5.6.1 Approximate prediction of hot zones

We have studied several simulations of the spatial heat pattern of randomly scattered point heat sources on the plane and found that there are two main

## Geometric Analysis of Hot Spots

---

factors that mainly cause a non-source point to get substantially hot. Either this point is located near a strong source or it is in the vicinity of a number of sources whose cumulative effect makes this point hot above the threshold. Performing exhaustive simulation at all points of an underlying grid, makes the analysis prohibitively slow. This has motivated us to propose a technique for prediction that would capture both the above aspects to some extent.

Voronoi Diagrams were traditionally known as the solution for *The Post Office Problem*. In an abstract sense, there is a set of central locations called sites on the plane that provide certain services. The question to be answered is if a person is located at point  $x$  on the plane, which is the site he should access to get services from. Generally, it is the site which is nearest from the point  $x$ , based on a suitable distance function. Voronoi Diagrams tessellate the plane into regions around each site, so that we can know which is the region of influence for any particular site.

Voronoi diagrams can be generalized in many ways [9]. One way of generalization is to consider point sets in higher-dimensional spaces. Another generalization concerns the distance metric that is used. It may be either the Manhattan metric ( $L_1$  metric) or the Euclidian metric ( $L_2$  metric) or in general the  $L_p$  metric [67]. One can also define a distance function by assigning weights to the sites. The resulting diagrams are called weighted Voronoi Diagrams and they are of two types – multiplicatively weighted and additively weighted [7]. Power diagrams [8] are another generalization of Voronoi diagrams where a different distance function is used. Sometimes no distance function is used altogether and the diagrams are defined in terms of bisectors between any two sites and they are called abstract Voronoi diagrams [60]. Instead of partitioning the space into regions according to the closest sites, one can also partition it according to the  $k$  closest sites, for some  $1 \leq k \leq n - 1$ . The diagrams obtained in this way are called higher-order Voronoi diagrams.

For a given  $k$ , the diagram is called the order- $k$  Voronoi diagram [21].

In our case, we are going to use order-1 multiplicatively weighted Voronoi Diagrams. Multiplicatively weighted Voronoi diagram is drawn by using a distance function  $d(p, p(i)) = \text{dis}(p, p(i))/w(i)$ , where  $\text{dis}$  is the Euclidean distance of a typical point  $p$  from the  $i$ th site  $p(i)$ , and  $w(i)$  is the weight of  $p(i)$ . Note that the edges in this type of diagram are generally *circular* unless the edge acts as a separation between two sites of identical weight, in which case it becomes a straight line as in an ordinary Voronoi Diagram. As the effective distance is obtained by dividing the Euclidean distance with the weight/strength of the site, the region belonging to a site with relatively higher weight will have relatively larger area. As for example, the separation between two sites  $p(1)$  and  $p(2)$  with weight  $w$  and  $2w$  respectively, will be the locus of a point  $p$  having the following property. If the distance of  $p$  from  $p(1)$  is  $d_1$  then its distance from  $p(2)$  is  $2d_1$ . This will ensure that  $d(p, p(1)) = d_1/w = 2d_1/2w = d(p, p(2))$ . Considering the context of *The Post Office Problem*, this makes sense as we expect a post office of larger size should serve a region of larger area. However for our thermal analysis problem, what we do is slightly different. Given the positions and strengths of the point heat sources, we define the weight of each source to be the inverse of its thermal strength. For avoiding numerical instability, we can ignore the sources that have very low value. After defining the weights, a multiplicatively weighted Voronoi diagram is drawn for these point sources with the weights defined as above. The effect of considering the inverses of the heat strengths is that the border of the Voronoi cells containing the sources of higher strength will be pulled inward, making the area of those cells relatively smaller. Our claim is that if we now consider the list of Voronoi cells in the ascending order of area occupied, most of the hot spots will lie in the cells that come in the beginning of the list. This will give a quick way to roughly know which are the hot zones on a chip.

## Geometric Analysis of Hot Spots

---

Consider Table 5.5 for a typical example with 11 thermal sources scattered over a floor of dimension 30 x 20. The first column gives the coordinates of the sources and the second column designates their corresponding raw heat strength. The next column gives the cumulative heat power at the source points evaluated as per *unit circle model* using Mathematica 5.0. Figure 5.8 gives the multiplicatively weighted Voronoi diagram for these 11 sources, the weight of each source being the inverse of the raw heat strength of that source. Column 4 of Table 5.5 shows the Voronoi cell corresponding to each source and the last column presents the percentage of area occupied by that cell on the floor. The results in Table 5.5 demonstrates that the area of a Voronoi cell is inversely related in a strong way with the cumulative heat received at the corresponding source points. The Voronoi cells in the ascending order of their area are H, B, D, I, F, G, K, E, C, A, and J. On the other hand the cells in the descending order of cumulative heat received at the corresponding source points are H, I, B, F, D, G, C, E, K, A, and J.

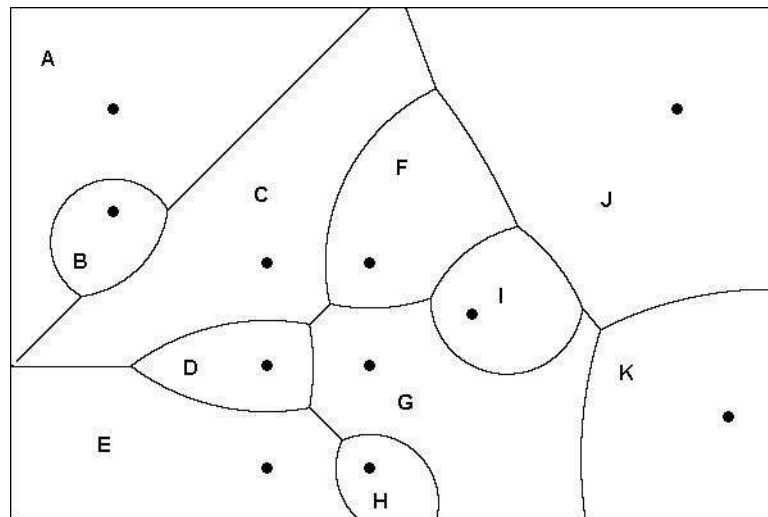


Figure 5.8: Weighted Voronoi diagram for 11 heat sources, based on raw strengths

In Figure 5.9, we show the relevant portion of the results obtained from

Table 5.5: Thermal data for 11 weighted sources

(x, y)	Raw strength	Cum. heat	Voronoi cell	% Area
(4, 4)	1.00	1.86	A	14.43
(4, 8)	2.10	2.89	B	2.51
(10, 10)	1.00	2.28	C	13.43
(10, 14)	1.25	2.56	D	3.15
(10, 18)	1.00	2.18	E	10.81
(14, 10)	1.30	2.58	F	6.67
(14, 14)	1.00	2.51	G	10.29
(14, 18)	2.00	3.05	H	1.82
(18, 12)	2.00	3.02	I	3.98
(26, 4)	1.00	1.56	J	22.32
(28, 16)	1.50	2.05	K	10.58

simulation of these 11 weighted sources following the *unit circle model*. The circular dots show the positions of the heat sources and the rectangular ones show the non-source points where the cumulative heat power has exceeded the value of 2.10 units. It is the highest raw strength possessed by any of the heat sources and we consider this as the threshold for our example. The idea is that the chip must have been designed to tackle at least the raw power generated by any of the sources, since the raw values can always be calculated at least approximately using functional simulation. The highest cumulative power reported is 3.17 units at a non-source point with coordinates (14, 17). Note that it is even higher than the hottest source point, which has a cumulative heat power of 3.05 units as shown in Table 5.5. Existence of a non-source point receiving the highest heat power, though counter-intuitive, is not unnatural [88]. Note that the points designated with rectangular marks have three different

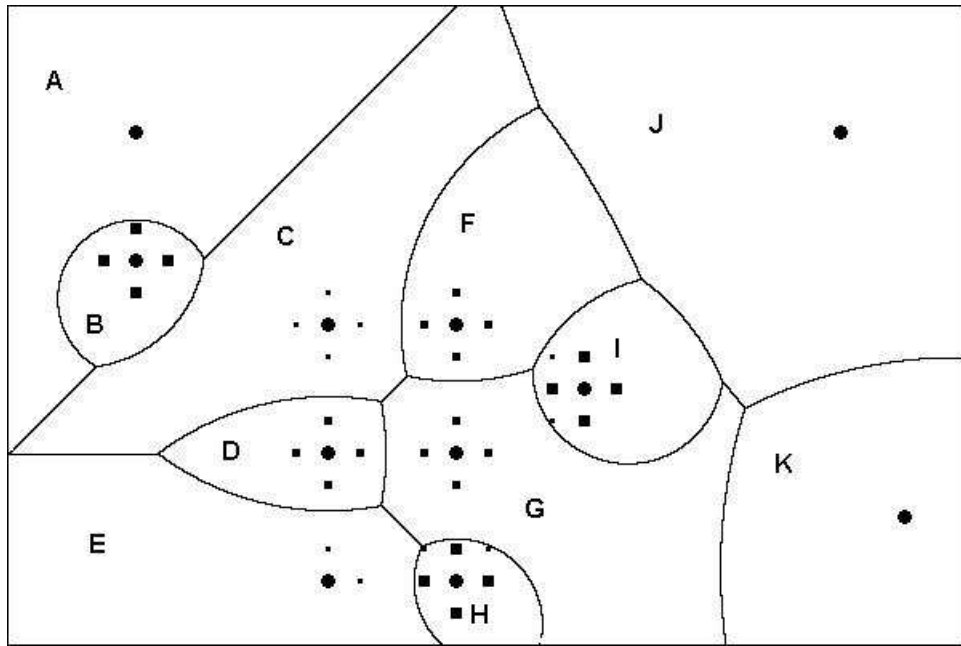


Figure 5.9: Superposition of simulation result and Voronoi diagram of 11 weighted heat sources

sizes proportional to their cumulative heat content. We have divided the range between 2.10 units to 3.17 units into three equal partitions. The points ( $\in P_U$ ) having the cumulative power in the upper half (2.81 units to 3.17 units) have the largest size. The points ( $\in P_M$ ) in the middle half (2.45 units to 2.81 units) have the medium size and the points ( $\in P_L$ ) in the lower half (2.10 units to 2.45 units) have the smallest size. Our investigation reveals that most of the hot points lie in the Voronoi cells having smaller area (critical cells). Thus, if we consider the 4 cells with smallest area, i.e., H, B, D and I, all the points in  $P_U$  lie within them. The sum total area of these 4 cells is however, only about 11.5% of the total area. If we further consider the next two cells in order of area, namely F and G, all the points in  $P_M$  lie either within them or in cell D that we have already considered. Hence, this provides an empirical verification of the claim that we made earlier.

Figure 5.10 gives the ordinary Voronoi diagram for 11 sources in the same

positions as Figure 5.8 but for a special case when all the sources have equal strengths. In other words, the sources are unweighted. As in the weighted case, in Figure 5.11 we show the results of simulation for these unweighted sources. Note that, here also the results are similar. The relatively hot points lie more in the Voronoi cells with smaller area. Table 5.6 presents the results of simulation as well as the area of the Voronoi cells for these unweighted sources in exactly same manner as Table 5.5.

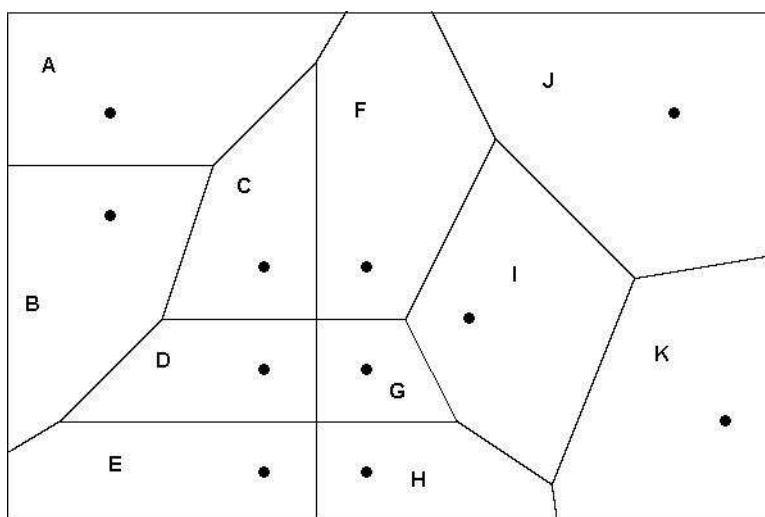


Figure 5.10: Ordinary Voronoi diagram for 11 heat sources of uniform weight

### 5.6.2 Local simulation on critical Voronoi cells

Having made the novel observation stated in the earlier subsection, we now propose a two-fold strategy. Given the source points with their raw strengths, we first draw the Voronoi diagram either ordinary or multiplicatively weighted depending upon whether or not the sources are uniformly weighted. The routine drawing the diagram should also report the areas corresponding to the Voronoi cells. Next we sort the areas in the ascending order. We then start with the cells with the smallest areas and do an exhaustive simulation at all

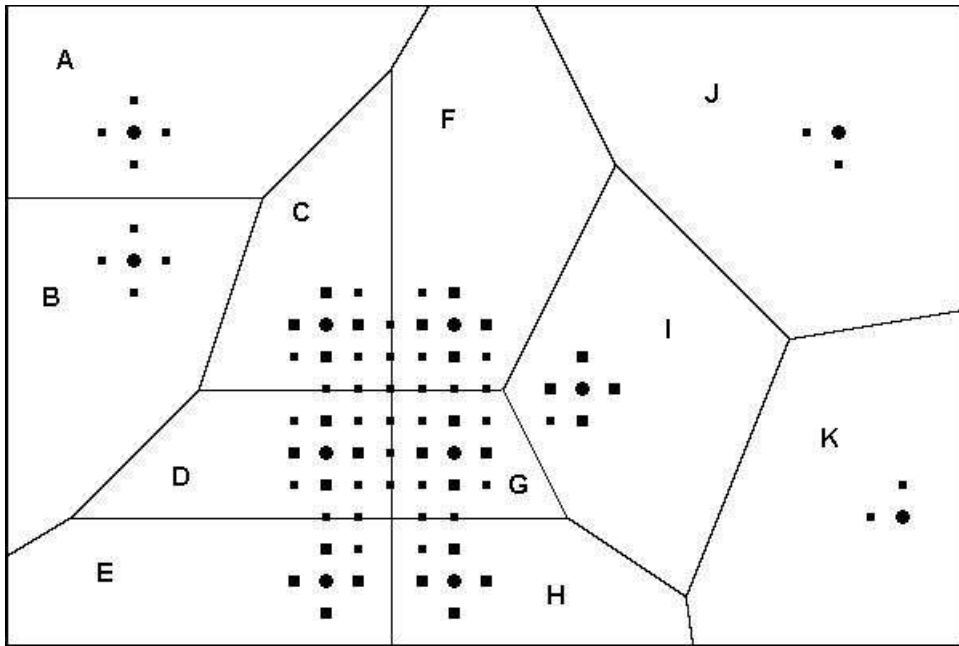


Figure 5.11: Superposition of simulation result and Voronoi diagram of 11 uniformly weighted heat sources

the grid points lying within that cell [88]. Our empirical observation is that simulating about 10 to 15 percent of the total area is sufficient to locate the hot spots or zones of interest. The time-complexity of the optimal algorithm for drawing ordinary Voronoi diagrams is  $O(n \lg n)$  and that of multiplicatively weighted ones is slightly higher [7]. The hybrid approach is found to provide a saving in simulation time by more than 85% on the average.

The program for drawing multiplicatively weighted Voronoi diagram was implemented using the Visual C language with graphics libraries. It was run on a pentium-based machine with 256 Mb RAM and for problems of moderate size, the run time was negligible compared to the simulation time.

Table 5.6: Thermal data for 11 unweighted sources

(x, y)	Raw strength	Cum. heat	Voronoi cell	% Area
(4, 4)	1.00	1.56	A	11.00
(4, 8)	1.00	1.65	B	9.91
(10, 10)	1.00	1.91	C	6.38
(10, 14)	1.00	1.99	D	5.35
(10, 18)	1.00	1.81	E	7.73
(14, 10)	1.00	1.94	F	10.73
(14, 14)	1.00	2.04	G	3.00
(14, 18)	1.00	1.84	H	5.33
(18, 12)	1.00	1.79	I	11.14
(26, 4)	1.00	1.39	J	17.26
(28, 16)	1.00	1.40	K	12.18

## 5.7 Dispersing the hot spots

Once we identify a zone of a chip as critically hot, the next challenge is to suggest some methodology that will help in cooling down those zones. Here, we assume that thermal violations have occurred even in the presence of appropriate number of thermal vias or heat sink mechanisms already placed on the board. If we want to decrease the strength of a heat source, it will require change at the circuit level, which may be quite difficult to do at an advanced stage of the design cycle. So we rather intend to dilate the clusters of hot spots on the floor that have crossed the thermal threshold, to ease dissipation. However, in doing so we do not want to disturb the adjacency (neighborhood) relationships of the different blocks of the design, which otherwise may change the floorplanning/placement significantly. In other words, geometrically we

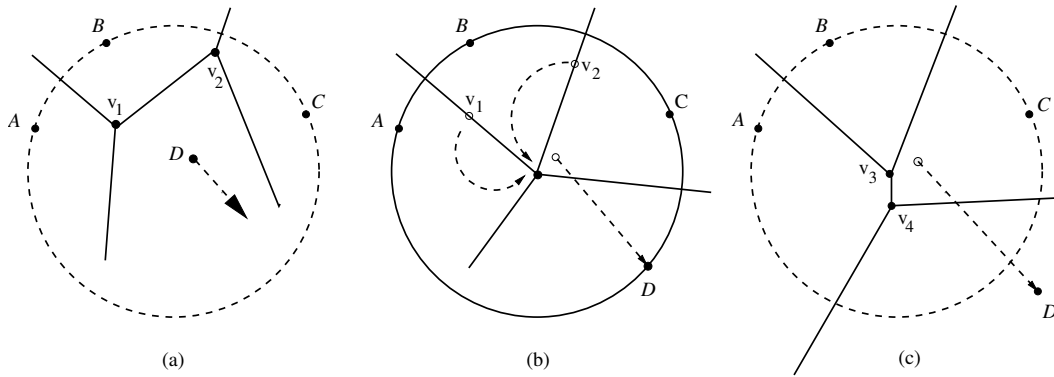


Figure 5.12: Moving source  $D$  in a portion of the floor that has crossed the thermal threshold

do not want to change the topology of the order-1 Voronoi diagram obtained from the point sources, ignoring their strengths. Thus, if we move one heat source away from the other heat sources close by we do not want to change the neighborhood relationship of that point source as captured by the Voronoi diagram. For every point heat source we will define a safe zone where it can be moved around without changing the topology of the Voronoi Diagram. For example, in Figure 5.12(a),  $A$ ,  $B$ ,  $C$  and  $D$  are thermal source points on the floor. If we start to move the point  $D$  away from the other points the Voronoi points  $v_1$  and  $v_2$  will start coming closer. This will continue till  $D$  reaches the circle passing through the points  $A$ ,  $B$  and  $C$ , when the points  $v_1$  and  $v_2$  will coincide, as shown in Figure 5.12(b). If  $D$  crosses the circle and goes to the other side, the neighborhood relationship of the Voronoi cells will change into that of Figure 5.12(c). Hence, as far as the segment  $(v_1, v_2)$  is concerned, the point  $D$  is free to move anywhere within the circle defined by  $A$ ,  $B$  and  $C$ . But if it becomes concyclic with those three points we end up in a degenerate case and when  $D$  crosses the boundary, the topology of the Voronoi diagram changes.

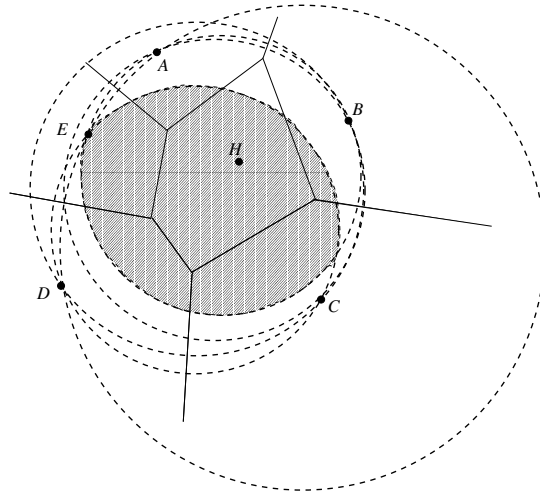


Figure 5.13: Safe zone of movement for source  $H$

### 5.7.1 Defining the safe zone

For any heat source point  $H$ , let us consider all the point sources in its neighboring Voronoi zones/cells. For each consecutive triplet among these neighboring point sources, we draw the circumscribing circle, as shown in Figure 5.13. If  $H$  has  $k$  neighbors then the number of circles will be  $k$ . Thus in Figure 5.13, where five point heat sources  $A$ ,  $B$ ,  $C$ ,  $D$ , and  $E$  surround  $H$  in the Voronoi diagram, the consecutive triplets will be  $(A, B, C)$ ,  $(B, C, D)$ ,  $(C, D, E)$ ,  $(D, E, A)$ , and  $(E, A, B)$ . The common intersection region (shaded) of these  $k$  circles is the safe zone of movement for the point source  $H$ . As described above, it is easy to verify that if  $H$  moves in the shaded region, the neighboring (adjacency) relationship among the Voronoi cells will remain invariant. Designing an algorithm to propose an alternative placement scheme by using the above observation is in the agenda of our future work.

### 5.8 Concluding remarks

We have proposed a number of models both in the continuous and discrete domain to model the thermal behavior of a VLSI chip. One important aspect we have observed in all the models is that there are zones in the chip which may become substantially hot even without containing any active heat source. We conclude that it may not be enough to guard only the active regions to make the chip thermally safe. First, we have presented some simulation based results to substantiate our claim. It also demonstrates how the hot zones of the chip can be identified using exhaustive simulation for time-invariant sources. Detailed analysis of time-varying sources is also in our agenda. Next, we propose a methodology that helps in predicting the thermal pattern of a chip without probing all the grid points. This is a two-fold approach to be used for fast identification of hot zones on a chip. For a set of points on the chip floor with their respective thermal strengths, we draw a multiplicatively weighted Voronoi diagram using the inverse of its thermal strength as the weight for each source. This in turn helps us to mark some zones of the chip to be under potential thermal threat. To make the prediction further accurate, we then run simulation in and around these critical regions only, to do the final prediction of hot spots/zones on the chip. We further propose a simple scheme to move some of the source points on the chip so that the thermal profiles of the critical zones go down below the threshold level. These movements are done preserving the Voronoi neighborhood of the sources. As Voronoi diagrams can be calculated by polynomial algorithms of low order, the first prediction scheme can be implemented very efficiently. As a result a significant saving in simulation time can be achieved while doing hot spot analysis.

# Chapter 6

## Density vs. Discrepancy of Points

### 6.1 Introduction and prior work

Thermal analysis has become one of the hot topics as power density in some regions of a VLSI chip is rising critically with ever-increasing clock-rate and device density. Proper placement of heat sinks and thermal vias have become issues of utmost importance. Some researchers have addressed the problem of hot spot identification [88] and others have proposed alternative placement schemes to cool down a hot chip [127]. Abstracting thermal sources as weighted points on the floor, the problem reduces to identifying the region with the highest concentration of points. Also if the region having minimum concentration of points is identified some source points from regions of higher concentration may be moved there. The motivation for our present research mainly comes from the above discussion.

Distribution of points on a plane has received considerable attention in the literature. It is quite popular in the domain of computational geometry and

## Density vs. Discrepancy of Points

---

is equally important for a number of practical applications. For a set  $S$  of scattered points on the plane, a question frequently arises - whether a cluster  $C (\subseteq S)$  is more concentrated or rarefied with respect to  $S$ . *Discrepancy measure* of a set of points tries to answer the above question [11, 6]. However, in the context of thermal analysis we preferred to introduce a new concept, called *density of the floor*. We prove several results that greatly reduce the search space for identifying the region with maximum (minimum) density. We also propose algorithms for solving some of these problems. Some related work either considers fixed number of points [111] or uses a fixed-size rectangle for calculating pattern density [65].

## 6.2 Problem formulation

Let  $F$  be a rectangular floor containing a set  $S$  of  $n$  points, such that no two points lie on the same horizontal (vertical) line. Each point  $p_i \in S$  is attached with a positive real weight  $w_i$ . We define *density* of an axis-parallel region  $R$  with area  $A(R)$  as  $\frac{\sum_{p_i \in R} w_i}{A(R)}$ . Note that, the weight of a point represents the relative strength of the corresponding thermal source. If all the sources are of uniform strength, the points will become unweighted and the *density* will reduce to  $\frac{\#(R)}{A(R)}$ ,  $\#(R)$  being the number of points in  $R$ . The normalized density for a set of unweighted points in  $R$  is  $\delta(R) = \frac{\#(R)}{A(R)} / \frac{n}{Area(F)}$ , so that  $\delta(F) = 1$ . Unless otherwise mentioned, all rectangles referred below are axis-parallel. We will consider the following two problems:

**Problem  $P_{min}$**  : Find the cluster of  $k(k \geq 2)$  points in  $S$ , such that the minimum area rectangle covering them attains the highest density on the floor  $R$ .

**Problem  $P_{max}$**  : Find the cluster of  $k(k \geq 1)$  points from  $S$ , such that the maximum area rectangle covering them attains the lowest density on the

bounded floor  $R$ .

### 6.3 Problem $P_{min}$

Note that we do not consider the maximum density of a region covering only one point as it can be made arbitrarily high. First, we prove that the maximum density occurs for a cluster  $C \subseteq S$  containing only two points.

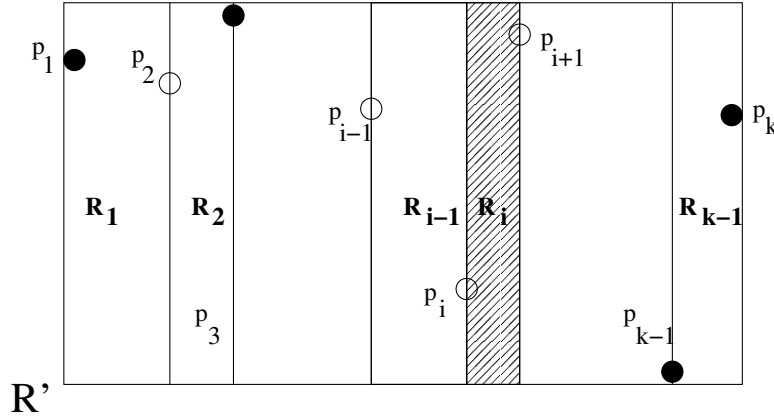


Figure 6.1: Smallest rectangle containing  $k$  points

**Lemma 6.1** *Let each point  $p_i \in S$  be attached with a positive weight  $w_i$  and there exist a cluster  $S' \subseteq S$  of  $k \geq 2$  points such that no two of them lie on the same horizontal (vertical) line. Then there exists a pair of points  $p_i, p_j \in S'$  such that the density of the smallest rectangle containing  $(p_i, p_j)$  is greater than the density of the smallest rectangle containing  $S'$ .*

**Proof.** Without loss of generality, we name the points in  $S'$  in increasing order of their  $x$ -coordinates and let the area of a rectangle  $R$  be represented by  $A(R)$ . Consider the smallest rectangle  $R'$  containing the point-set  $S'$ . Split  $R'$  into  $k - 1$  vertical strips by drawing vertical lines through each point. The rectangular strip whose vertical sides passes through  $p_i$  and  $p_{i+1}$  is named as

## Density vs. Discrepancy of Points

---

$R_i$ . The rectangle shown in Fig. 6.1 represents  $R'$  covering  $k$  points of  $S'$ . Now, the density of the entire floor is

$$\begin{aligned} \frac{\sum_{i=1}^k w_i}{A(R')} &= \frac{\sum_{i=1}^k w_i}{\sum_{i=1}^{k-1} A(R_i)} \leq \frac{w_1+2 \sum_{i=2}^{k-1} w_i+w_k}{\sum_{i=1}^{k-1} A(R_i)} \\ &= \frac{(w_1+w_2)+(w_2+w_3)+\dots+(w_{k-1}+w_k)}{A(R_1)+A(R_2)+\dots+A(R_{k-1})} \leq \text{Max}_{i=1}^{k-1} \frac{(w_i+w_{i+1})}{A(R_i)} \quad (\text{by Fact 6.1}). \end{aligned}$$

Again the smallest rectangle containing the concerned pair may have a higher density, as it may not span the entire strip. Hence the result.  $\square$

**Fact 6.1** *If a quantity  $Q = \frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i}$ ,  $a_i, b_i > 0$ , then  $\text{Min}_{i=1}^n \frac{a_i}{b_i} \leq Q \leq \text{Max}_{i=1}^n \frac{a_i}{b_i}$ .*

**Proof.** Let  $\frac{a_i}{b_i}$  attain the minimum and maximum value for  $i = *$  and  $i = \#$  respectively.

$$\text{For } 1 \leq i \leq n, \frac{a_*}{b_*} \leq \frac{a_i}{b_i} \leq \frac{a_\#}{b_\#} \Rightarrow \frac{a_*}{b_*} b_i \leq a_i \leq \frac{a_\#}{b_\#} b_i. \quad (\text{as } b_i > 0)$$

Adding over  $i$ , we get,

$$\frac{a_*}{b_*} \sum_{i=1}^n b_i \leq \sum_{i=1}^n a_i \leq \frac{a_\#}{b_\#} \sum_{i=1}^n b_i \Rightarrow \frac{a_*}{b_*} \leq \frac{\sum_{i=1}^n a_i}{\sum_{i=1}^n b_i} \leq \frac{a_\#}{b_\#}.$$

Hence the result.  $\square$

Note that, by the above lemma in the unweighted case,  $P_{min}$  gets reduced to the problem of finding the pair of points such that the area of the rectangle having those two points at its diagonally opposite corners is minimum. This can be efficiently solved in polynomial time.

### 6.3.1 Algorithm

We now discuss on an algorithm for finding out the smallest area rectangle covering exactly two points from the set  $S$ . The simplest way is to consider each pair of points, and compute the density of the rectangle whose diagonal is defined by these two points in  $O(n^2)$  time. It works for weighted case also. We show below that the time complexity can be improved for the unweighted case.

Note that, a rectangle achieving the maximum density is a *corner rectangle* [3]. Thus, the problem reduces to identifying a corner rectangle having minimum area. An  $O(n \log^2 n)$  time and  $O(n)$  space algorithm finds out the largest corner rectangle using the technique of monotone matrix searching [3], and we show that it works for our problem also. This algorithm uses a two-level divide and conquer strategy. It first splits the point set into two almost equal halves using a horizontal line  $H$  and then by a vertical line  $V$ . They intersect at  $O$  and split the plane into four quadrants. In each quadrant  $i$ , we identify an axis-parallel staircase  $S_i$  around the point  $O$  (Fig. 6.2(a)). The convex corners of this staircase contains a member in  $S$  and the interior of the axis-parallel polygon bounded by  $S_i$ ,  $x$ -axis and  $y$ -axis is empty. Consider the staircases in second and fourth quadrants. A rectangle with top-left and bottom-right corner at the convex corner of these two staircases respectively does not contain any point of second and fourth quadrant. If we consider a matrix  $M$  whose rows and columns correspond to the points in second and fourth quadrants respectively, and its each entry correspond to the area of a rectangle defined by one point from each of these staircases at the end of its diagonal, it will be a monotone matrix. This algorithm uses a monotone matrix w.r.t.  $\geq$  sign [4], whereas for our case we need a monotone matrix w.r.t.  $\leq$  sign.

**Definition 6.1** *A matrix  $A$  is said to be monotone (with respect to  $\leq$  sign) if for every  $j, k, j', k'$  with  $j < j', k < k'$ , if  $A[j, k] \leq A[j, k']$  then  $A[j', k] \leq A[j', k']$ .*

The maximum-valued element in an  $n \times n$  monotone matrix can be found in  $O(n)$  time [4] and hence in our case we can find the minimum element also in  $O(n)$  time. Note that, some of these rectangles may contain point(s) of first and/or third quadrant, needing an  $O(n)$  time clean-up procedure [20]. This is followed by two recursive steps one in the vertical direction and the other in the horizontal direction, each contributing a  $\log n$  factor to the final complexity of

## Density vs. Discrepancy of Points

$O(n \log^2 n)$ , and it is applicable identically to our problem. In order to search the smallest corner rectangle, we arrange the elements of the matrix such that its rows ( $U_i$ s) are ordered as in the earlier problem, but columns ( $V_i$ s) are ordered in the reverse direction as shown in Fig. 6.2(a).

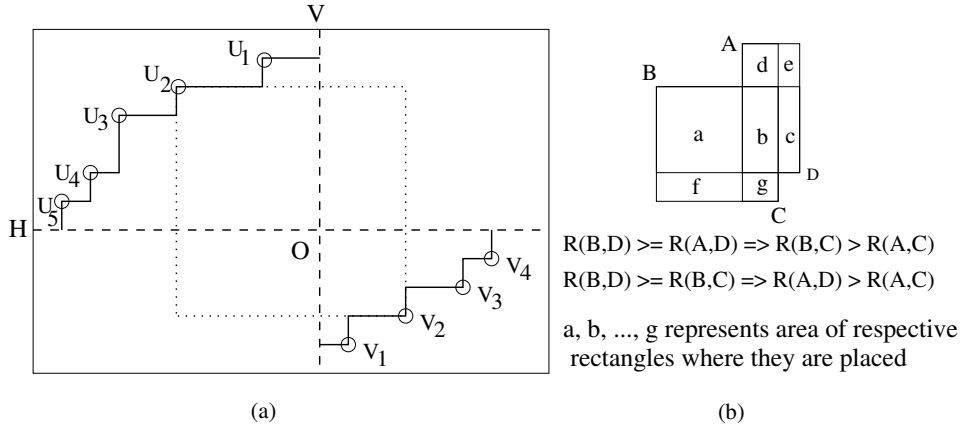


Figure 6.2: Rectangles with corners in opposite quadrants

Facts 6.2 and 6.3 below establish that it is a monotone matrix with respect to  $\leq$  sign. Consider Fig. 6.2(b). Here, a rectangle defined by a pair of points  $(p, q)$  as its diagonal, is denoted by  $R(p, q)$ .

**Fact 6.2**  $R(B, D) \geq R(A, D) \Rightarrow R(B, C) > R(A, C)$ .

**Proof.**  $R(B, D) \geq R(A, D) \Rightarrow a + b + c \geq b + c + d + e$

$$\Rightarrow a + b \geq b + d + e \Rightarrow a + b + f + g \geq b + c + d + e + f + g$$

$$\Rightarrow a + b + f + g > b + d + g \Rightarrow R(B, C) > R(A, C). \quad \square$$

**Fact 6.3**  $R(B, D) \geq R(B, C) \Rightarrow R(A, D) > R(A, C)$ .

**Proof.**  $R(B, D) \geq R(B, C) \Rightarrow a + b + c \geq a + b + f + g$

$$\Rightarrow b + c \geq b + f + g \Rightarrow b + c + d + e \geq b + f + g + d + e$$

$$\Rightarrow b + c + d + e > b + d + g \Rightarrow R(A, D) > R(A, C). \quad \square$$

Hence the complexity of finding the smallest corner rectangle is also  $O(n \log^2 n)$ .

### 6.3.2 Density in general case

In the isothetic domain, thin rectangles pose problems as their densities become unnecessarily high, which may not properly reflect the real situation. However, similar results are possible beyond the isothetic domain also. For general case, we revise the definition of density for a set of unweighted points as follows:

**Definition 6.2** *Let  $S' \subseteq S$  be a subset of points. The density of  $S'$  is defined as  $\delta(S') = \frac{|S'|}{A(\text{CONV}(S'))}$ , where  $|S'|$  indicates the cardinality of  $S'$ , and  $\text{CONV}(S')$  indicates the convex hull of the point set  $S'$ .*

Assuming that no three points in  $S$  are colinear, Fact 6.4 says that the maximum density under this definition occurs for a triple of points.

**Fact 6.4** *Let there be a set  $S$  of  $n > 2$  points on a rectangular floor  $R$  such that no three of those points are co-linear. There exists a cluster  $C \subseteq S$  of three points such that the triangle having those three points as its vertices and covering no other point from the set  $S$  achieves the highest density on the floor.*

**Proof.** Consider a triangulation of any subset  $S' \subseteq S$  (Fig. 6.3(a)). We show that there exists a triangle with higher density (by Def. 6.2) than that of  $S'$ . Let  $|S'| = k$ . The number of triangles in any triangulation of  $S'$  is  $\geq k - 2$ , minimum occurs when all the  $k$  points lie on the convex hull (Fig. 6.3(b)). Let the area of the convex hull be  $A$ . As average value lies between minimum and maximum, there exists a triangle  $\tau$  with area  $A(\tau) \leq A/(k - 2)$ . Now,  $\delta(S') = \frac{k}{A}$  and  $\delta(\tau) \geq \frac{3}{A/(k-2)}$ . As  $3(k-2) \geq k$ , for  $k \geq 3$ , the result follows.  $\square$

## 6.4 Problem $P_{max}$

We now consider the problem of identifying a rectangle on the floor having minimum non-zero density. Given a bounded floor containing a set  $S$  of points,

## Density vs. Discrepancy of Points

---

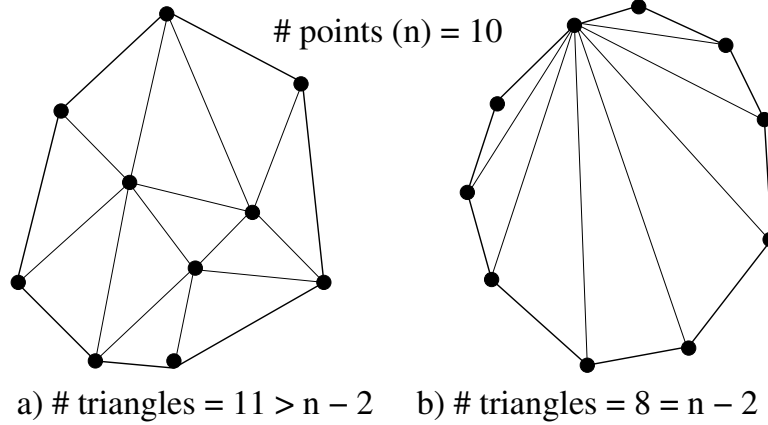


Figure 6.3: Density in a non-isothetic scenario

the following lemma says that the minimum density occurs for a rectangle containing a single point in  $S$ .

**Lemma 6.2** *Let each point  $p_i \in S$  be attached with a weight  $w_i > 0$ , and there exist a cluster  $S' \subseteq S$  of  $k \geq 1$  point(s). Then there exists a point  $p_i \in S'$  such that the density of the largest rectangle that contains only point  $p_i$  is less than the density of the largest rectangle  $R'$  containing  $S'$  and no other points from  $S$ .*

**Proof.** Without loss of generality, consider the points in  $S'$  in increasing order of their  $x$ -coordinates and let  $A(R)$  be the area of rectangle  $R$ . The rectangle shown in Fig. 6.4 represents  $R'$  covering  $k$  points of  $S'$ . It is bounded by points that belong to  $S$  but not to  $S'$ . Split  $R'$  into  $k + 1$  vertical strips by drawing vertical lines through each point. The vertical sides of strip  $R_i$  passes through  $p_i$  and  $p_{i+1}$ . Now,

$$\begin{aligned} \delta(R') &= \frac{\sum_{i=1}^k w_i}{A(R')} = \frac{\sum_{i=1}^k w_i}{\sum_{i=0}^k A(R_i)} \leq \frac{\sum_{i=1}^k w_i}{A(R_0) + 2 \sum_{i=1}^{k-1} A(R_i) + A(R_k)} \\ &= \frac{w_1 + w_2 + \dots + w_k}{(A(R_0) + A(R_1)) + (A(R_1) + A(R_2)) + \dots + (A(R_{k-1}) + A(R_k))} \\ &\geq \text{Min}_{i=1}^k \frac{w_i}{(A(R_{i-1}) + A(R_i))} \quad (\text{by Fact 6.1}). \end{aligned}$$

Again the density of the largest rectangle containing the concerned point may

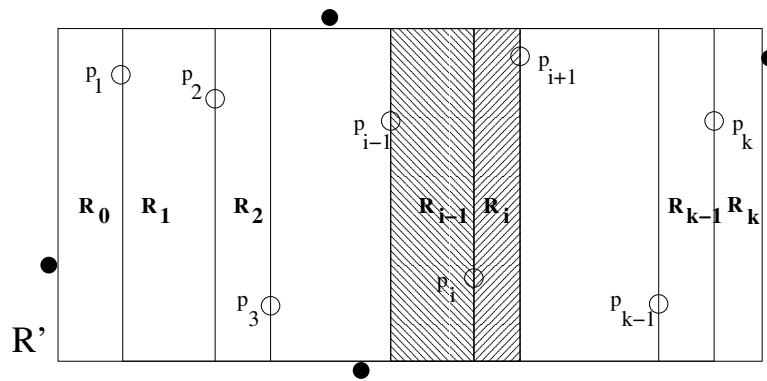


Figure 6.4: Largest rectangle containing  $k$  points

be less, as it may be possible to extend the shaded portion in Fig. 6.4 along the vertical direction. □

If  $R(p)$  denotes the largest rectangle containing only  $p \in S$  and no other point in  $S$ , then by the above lemma, for the unweighted case we just need to identify  $p_i \in S$  such that  $R(p_i)$  is largest among all the points.

## 6.5 Density vs. discrepancy

Assuming that the area containing a point set  $S$  is a unit square, the *discrepancy measure* of  $S$  may be defined as  $\Delta = \max_{r \in R} |area(r) - \frac{\#r}{n}|$ , where  $r$  is an arbitrary rectangular area and  $\#r$  indicates the number of points of  $S$  lying inside  $r$ . For any rectangle  $r$ , we introduce a concept of local discrepancy  $\Delta_l(r) = |area(r) - \frac{\#r}{n}|$ . In Fig. 6.5,  $\Delta_l(r) = |(0.5 * 0.25) - (3/10)| = 0.175$ . Let us try to interpret the main result of Section 3 from the definition of Discrepancy. We choose any three points on the rectangular floor  $R$  whose coordinates do not match along either axis. Let  $R_3$  be the skin-tight rectangle covering only those three points from  $S$ . So  $\Delta_l(A_3) = |A_3 - 3/n|$ , where  $A_3 = Area(R_3)$  and density  $D_3 = 3/A_3$ . From Section 3, we can say that there must exist a pair of points out of these 3 points covered by

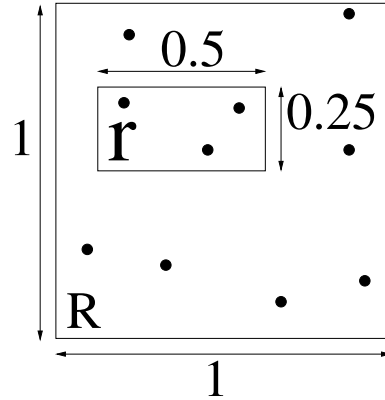


Figure 6.5: Local discrepancy of a set of points

the skin-tight isothetic rectangle  $R_2$  such that their density  $D_2 > D_3$ . Now,  $D_2 > D_3 \Rightarrow 2/A_2 > 3/A_3 \Rightarrow A_3 > 3/2A_2 \Rightarrow (A_3 - 3/n) > (3/2A_2 - 3/n)$ .

Hence  $(A_3 - 3/n) > 3/2(A_2 - 2/n)$ . This result leads to three possible cases.

*Case 1:*  $A_2 - 2/n$  is +ve  $\Rightarrow A_3 - 3/n$  is +ve.

*Case 2:*  $A_3 - 3/n$  is -ve  $\Rightarrow A_2 - 2/n$  is -ve.

*Case 3:*  $A_3 - 3/n$  is +ve but  $A_2 - 2/n$  is -ve.

Note that out of the three cases only in *Case 1* it is guaranteed that  $|(A_3 - 3/n)| > |(A_2 - 2/n)|$ , for the other two cases any of those quantities may be greater or smaller. Hence nothing can be concluded by comparing the values of the local discrepancy of three points and that of any pair chosen from those 3 points.

Note that, two set of points having the same density may have their local discrepancy different and conversely. However the main problem is not caused by them. The fact that the rise or fall of density function does not map monotonically to the rise or fall of discrepancy function is the main deterrent in interpreting the results of one domain from the other.

Let us take a look at the expression for discrepancy. Considering an isothetic rectangular region  $r$  within  $R$ , if we substitute  $\alpha$  for  $\text{area}(r)/\text{area}(R)$  and  $\beta$  for  $\#r/n$ ,  $\Delta_i(r) = |\alpha - \beta|$ , where  $0 < \alpha, \beta \leq 1$ . The normalized density

$$\delta(r) = (\#r/area(r))/(n/area(R)) = (\#r/n)/(area(r)/area(R)) = \beta/\alpha.$$

Now we can look at the differences between density and discrepancy. Consider a case where  $n = 10$  points are scattered on a 1x1 rectangular floor such that two different subsets of those points  $S_1$  and  $S_2$  respectively has  $\alpha_1 = 0.3$ ,  $\beta_1 = 0.4$  and  $\alpha_2 = 0.5$ ,  $\beta_2 = 0.6$ . So for  $S_1$ , 4 out of the 10 points are scattered in a skin-tight rectangle whose area is 30% of the whole floor and for  $S_2$ , 6 out of 10 points are scattered in an area of 0.5 units. The discrepancies of both these sets is same  $|0.3 - 0.4| = |0.5 - 0.6| = 0.1$ . However  $S_1$  has a density of  $4/3$  and  $S_2$  has a density of  $6/5$ , which are different. Next consider a case with similar set-up where  $\alpha_1 = 0.8$ ,  $\beta_1 = 0.6$  and  $\alpha_2 = 0.4$ ,  $\beta_2 = 0.3$ . Here density of both the sets is  $3/4$  but the discrepancies are respectively 0.2 and 0.1.

Finally we cite here two examples, where in the first one, increase in density causes discrepancy to rise, and in the second one, decrease in density causes discrepancy to rise. Consider again two set of points  $S_1$  and  $S_2$  with  $\alpha_1 = 0.5$ ,  $\beta_1 = 0.6$  and  $\alpha_2 = 0.4$ ,  $\beta_2 = 0.6$ . For  $S_1$ , density =  $6/5$  and discrepancy is 0.1, whereas for  $S_2$ , density =  $6/4$  (i.e. density increases) and discrepancy = 0.2 (i.e. discrepancy also increases). Now consider two set of points  $S_3$  and  $S_4$  with  $\alpha_3 = 0.5$ ,  $\beta_3 = 0.4$  and  $\alpha_4 = 0.6$ ,  $\beta_4 = 0.4$ . For  $S_3$ , density =  $4/5$  and discrepancy is 0.1, whereas for  $S_4$  density =  $4/6$  (i.e. density decreases) but discrepancy = 0.2 (i.e. discrepancy increases). So by tracking the change in density it is difficult to predict the change in discrepancy and vice versa.

## 6.6 Conclusion and future works

We have introduced a new concept for calculating density of points scattered in a plane. We also made some simple but novel observations that considerably reduces the search space for finding out the zones having maximum or minimum density. Then we have proposed some algorithms based on existing

## Density vs. Discrepancy of Points

---

geometric techniques to solve these problems. The whole technique may be applied for facilitating thermal analysis of ULSI chips. We have compared it with an existing concept of discrepancy and found that the results regarding density cannot be easily translated to the domain of discrepancy. Hence the concept of density may have some contribution on its own to interpret the properties of the relative distribution of a set of points scattered on a rectangular floor. It may be worthwhile to look into the density problem for higher dimensions. Either similar theorems can be proven or the results may be different from 2 dimension. Also the algorithms may not turn out to be as simple as 2 dimension.

# Chapter 7

## Classification of Path-delay Faults Using Stuck-at Faults

### 7.1 Introduction

Many physical design issues, for example, power grid layout and interconnect/repeater management, have strong influence on the timing behavior of the chip. When several cells drawing power ( $V_{dd}$ ) from the power grid through a common power via, switch simultaneously, a voltage drop (commonly known as power droop) may occur at the corresponding via. This may cause a slow-to-rise transition at the output of victim cells driven by the via [97]. Similarly, interconnect layout strategy is affected by availability of room on the silicon floor to place the required number of repeaters in order to reduce delay and to minimize skew [110, 106]. Correctness of such layout designs can only be verified by running/simulating delay tests [73]. Moreover, in nanometer design, on-chip inductive drop ( $Ldi/dt$ ) along the power grid cannot be ignored as the frequency is in the range of several GHz. Also, power supply voltage is getting scaled down with technology causing notable decrease in noise margin.

A change of 1% in power supply voltage can cause nearly 4% change in delay for most 90 nm static CMOS gates [125]. Thus modeling, analysis, and testing of transition/delay faults [63, 77], are extremely important to a chip designer.

Identification of untestable path-delay faults has applications to delay testing, timing analysis, and timing optimization of digital circuits. In this chapter, we establish a strong correlation between *false timing paths* and redundant stuck-at faults in an unfolded circuit obtained from the original circuit through simple transformations. Path-delay testing has turned out to be the most accurate form of delay testing [63, 73, 115]. The importance of delay faults is further strengthened by the recent observation that many defects in the deep-submicron regime cannot be modeled by the classical stuck-at faults [5], and that many spot defects like opens often pass test screens [101]. These *opens* usually manifest as timing faults and improved techniques of delay fault testing are needed [62]. In other words, the target of Delay Fault Testing and improved physical design is the same – generate efficient systems that can withstand the emerging challenges of high performance computing of the next decade.

The number of paths can be an exponential function of the number of lines in the circuit. However, all paths need not be tested as many paths may be incapable of producing timing errors in spite of their large propagation delays [104]. These are called *false paths*. Much work has been done on fast identification of false paths in combinational circuits [48, 116]. Their identification is equally important in sequential circuits [70, 124].

Majumder *et al.* explored the possibility of identifying untestable path-delay faults from the knowledge of redundant stuck-at faults in the circuit [83]. It was observed that while some untestable delay faults can be easily predicted from the list of redundant stuck-at faults, other untestable paths do not correspond to any redundant stuck-at fault. In fact, there exist many irredundant

circuits with untestable path delay faults.

Several authors have suggested construction of modified circuits, which preserve the original set of paths and contain redundant stuck-at faults on all untestable paths. Lam *et al.* [66] constructed an expanded form of circuit graph known as *leaf-dag* in which fanouts are permitted only at the input nodes. This is done by moving all fanouts and all inverters toward the primary inputs. The size of the leaf-dag may be exponential in the size of the circuit. The leaf-dag is used to identify the robust-dependent paths, which correspond to redundant stuck-at faults. Gharaybeh *et al.* [42] produced a two-level circuit where each original path is explicitly represented. They used the circuit to generate tests for singly-testable and multiply-testable paths. Redundant stuck-at faults identify the untestable paths.

In another approach, Gharaybeh *et al.* [43, 42] used simulation to split circuit nodes for separating tested and untested paths. The split circuit contains new redundant stuck-at faults, which correspond to the untestable paths.

In this chapter, we establish a one-to-one correspondence between every untestable path in a combinational circuit  $C$  and a redundant stuck-at fault in a circuit  $U$  obtained by unfolding  $C$  according to certain rules. For several well-known path-delay fault classifications, we then characterize each class in terms of testability of stuck-at faults in the unfolded circuit.

## 7.2 Circuit unfolding

In order to establish a one-to-one relation between every path in a circuit  $C$  and some stuck-at fault, we unfold the circuit from each fan-out point except the primary input fanouts. For every fanout in the circuit other than the fanouts of primary inputs, we duplicate the subcircuit feeding the fanout point so that

## Classification of Path-delay Faults Using Stuck-at Faults

---

each gate receiving inputs from that fanout point has a separate subcircuit connected to its input. We apply unfolding in this way until all gates in the circuit have single fanout. In the worst case, this might require an exponential complexity but still we can do it at least theoretically for any circuit. However, this expansion of circuit is just an abstraction that is used to prove our claims. Figure 7.1 shows the original circuit  $C$  and the unfolded circuit  $U$  for an exclusive-or function. It may be noted that for every path present in  $C$ , there is a corresponding path in  $U$  and vice versa. Further, every path in  $U$  uniquely corresponds to a primary input (or one of its fanout branches in the case it has multiple fanouts) and vice versa as each of the gates in  $U$  has only a single fanout. With every path  $P$  in  $C$  we associate two stuck-at faults on the primary input  $I_P$  (or possibly a fanout branch) where the corresponding path  $P_U$  in  $U$  originates. For a rising (falling) transition at the input of  $P$  we consider the stuck-at-0 (stuck-at-1) fault on  $I_P$  as done by Majhi *et al.* [79] and Majumder *et al.* [83].

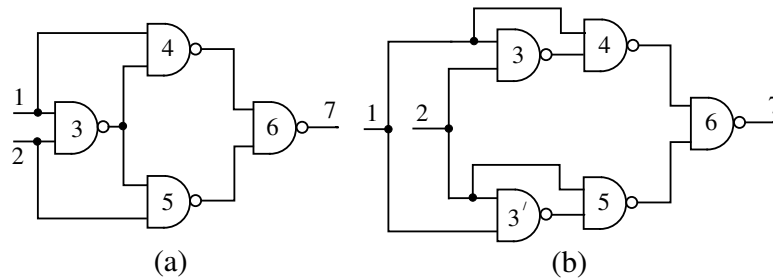


Figure 7.1: (a) Original circuit  $C$  (b) Unfolded circuit  $U$

An unfolded circuit is equivalent to leaf-dag [66, 109]. However, leaf-dag requires an additional step of preprocessing for moving the inversions to the input side, and changing the gate types by De-Morgan's law. It can be proved that a path-delay fault is testable if a corresponding stuck-at fault is testable. The work on leaf-dag did not address the issue of interpreting various fault classifications; in fact, classifications were not known at that time. In our

case, we allow the use of all simple gates in  $U$  that were present in the original circuit  $C$ .

Gharaybeh *et al.* [42] presented a tight “if and only if” correspondence between stuck-at fault and delay fault testabilities. However, their two-level equivalent circuits are complex involving timing parameters. Majhi *et al.* [79] and Majumder *et al.* [83] proved that certain redundant stuck-at faults imply untestability of some path-delay faults, but nothing was said about the converse. Sparmann and Koeller [117] considered the classification done by Lam *et al.* [66]. They proved that if a path  $P$  in a circuit is internally fanout free, and is non-robustly untestable, then a single stuck-at fault at the input line of  $P$  is redundant. Thus, their focus was on the converse case, which is used to remove false paths by eliminating redundant stuck-at faults. However, no general result on the “if and only if” condition between redundancy of stuck faults and untestability of path-delay faults, in a simple form, seems to have been known. The present work makes the following contributions:

- “If and only if” conditions are presented using a simple equivalent circuit.
- Multiply-testable paths (defined later) are described in terms of multiple stuck-at faults for the first time.
- Various delay fault classification schemes like robust-testable, functional sensitizable (FS) [23], single testable (ST), multiply testable (MT) [42, 41], primitive single path delay fault (SPDF), primitive multi-path delay fault (MPDF) [114] are interpreted using the stuck-at fault model alone, on the same equivalent circuit.
- A unified and complete characterization of path-delay faults is presented in terms of stuck-at faults.

## 7.3 Stuck-at faults in unfolded circuit and path classifications

Path-delay faults have been divided into different classes by various groups working in delay fault testing. The latest summary appears in Figure 7.2 [41, 80, 114]. The definitions and meaning of abbreviations are explained later.

Lam et al. (IEEE-TCAD '95)	Non-RD		RD	
Cheng and Chen (IEEE-TCAD '96)	Robust testable	Non-robust testable	Functional sensitizable	Redundant
Gharaybeh et al. (JETTA '97)	Singly-testable		Multiply-testable	Singly-testable dependent
Sivaraman and Strojwas (VLSI Design '97)	Primitive SPDF		Primitive MPDF	Primitive dependent

Figure 7.2: Classifications of path-delay faults

*Gharaybeh et al.'s classification.* [42] In this classification, paths are grouped into three categories: *singly-testable*, *multiply-testable* and *ST-dependent*.

**Definition 7.1** (Functional) *A path is singly-testable (ST) for a delay fault if and only if a test guarantees detection, given that it is the only fault present in the circuit [42]. An ST path is also called a non-robustly testable path.*

**Definition 7.2** (Structural) *A path is singly-testable (ST) for a delay fault if and only if there exists a vector-pair  $\langle v_1, v_2 \rangle$  such that: (i) a transition with appropriate direction is applied at the path input, and (ii) all side-inputs of the path are non-controlling in  $v_2$ .*

In the literature, the same conditions define a non-robust test for a path.

By definition, a ST path must have a non-robust delay test. However, it is possible to have testable paths that are not ST. Some singly-untestable paths can be tested in groups, i.e., a test can be found when several paths are simultaneously faulty. We call a group *minimal* if removal of any single path leaves the group untestable. Consider the following cases for a singly-untestable path  $A$ :

1. If any testable minimal group of paths that contains  $A$ , also contains at least one ST path, then  $A$  is called a *singly-testable-dependent (ST-dependent)* path [42]. We conclude that  $A$  need not be tested since it will not cause an incorrect operation in a circuit that has passed the tests for all ST paths.
2. If none of the testable groups of paths that contain  $A$ , includes any ST path, then  $A$  is called a *multiply-testable (MT)* path.

**Definition 7.3** (Functional) *A path is **multiply-testable (MT)** if and only if a test exists for a set of simultaneously faulty paths, none of which is ST [41].*

**Definition 7.4** (Structural) *A path is **multiply-testable (MT)** if and only if for a set of singly-untestable paths including the target MT path, there exists a pattern-pair  $\langle v_1, v_2 \rangle$  such that: (i) transitions with appropriate directions are applied at the primary inputs of the component paths, and (ii) all side-inputs to the set of paths are non-controlling under  $v_2$ .*

Let us consider the circuit of Figure 7.1 again. A *physical path* is a continuous sequence of gates and lines starting from a primary input to some primary output. A *logical path* is a physical path with a transition direction  $\uparrow$ (rising) or  $\downarrow$ (falling) at its input. Out of the twelve logical paths in this circuit, only two paths,  $\downarrow 1 - 3 - 4 - 6 - 7$  and  $\downarrow 2 - 3 - 5 - 6 - 7$ , are ST-dependent.

## Classification of Path-delay Faults Using Stuck-at Faults

---

All other paths are ST. In the unfolded circuit, the corresponding paths are  $\downarrow 1 - 3 - 4 - 6 - 7$  and  $\downarrow 2 - 3' - 5 - 6 - 7$ . The stuck-at-1 faults on the fanout branches from input 1 to gate 3 (henceforth denoted as  $1 - 3$ ) and on  $2 - 3'$  are redundant as shown in Figure 7.3. Rest of the ten stuck-at faults are all testable. The two redundant stuck-at faults are of the type *undrivable* (a fault that cannot be excited and propagated to observable outputs simultaneously). Majumder *et al.* correctly predicted the untestability of the two paths specified above in the unfolded circuit [83]. Here, we establish the fact that testability of any path-delay fault is strongly related to the detectability of some stuck-at faults in the unfolded circuit. For a path-delay fault in a multi-output circuit detectability of a stuck-at fault should be considered at the corresponding output.

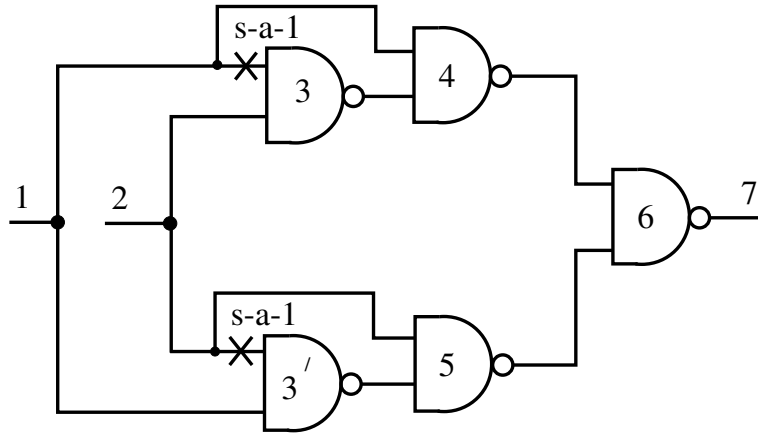


Figure 7.3: Singly-testable paths

**Theorem 7.1** *A path-delay fault  $\uparrow P$  ( $\downarrow P$ ) in a circuit is ST if and only if the stuck-at-0(1) fault at the fanout branch of the input or the input line itself (if it has no fanout) of the corresponding path in the unfolded circuit is testable.*

**Proof.** Suppose  $\uparrow P$  is testable. Then by definition, there exists a vector pair  $\langle v_1, v_2 \rangle$  that applies a rising transition at the input of  $P$  and all side-inputs

of the path are non-controlling under  $v_2$ . Unfolding keeps the topology of the circuit unchanged, although duplication of some portions may cause possible logical redundancies. As the unfolded circuit  $U$  preserves the function of the original circuit, under the vector  $v_2$ , the stuck-at-0 fault at the input fanout branch of  $P$  will be excited with logic 1 both in  $C$  and  $U$ . As non-controlling values appear at all side-inputs of  $P$ , the required stuck-at fault will be testable in  $U$ .

If  $\uparrow P$  is untestable, then a vector pair that produces a rising transition at the input of  $P$  fails to statically sensitize the path  $P$  under  $v_2$ . In other words, the corresponding stuck-at-0 fault in  $U$  will be undrivable redundant. This happens because the stuck-at fault of our interest is observable only along path  $P$  and not through any other path. Thus in Figure 7.1, though  $C$  was stuck-at fault irredundant,  $U$  has two redundant stuck-at faults as the observability of the faults becomes more restricted in  $U$ . The proof for  $\downarrow P$  follows easily.  $\square$

Next, we derive a similar result for MT paths.

**Theorem 7.2** *A set of logical paths  $S$  in a circuit is MT if and only if in the unfolded circuit  $U$  the multiple stuck-at fault (constituting the single stuck-at faults corresponding to the paths in  $S$ ) is testable. Each of these single stuck-at faults must be individually redundant.*

**Proof.** Let the set of paths  $S$  be MT. Hence each path in  $S$  is singly untestable by definition. By theorem 7.1 each of the corresponding stuck-at faults in  $U$  is redundant. Now, there exists a vector pair  $\langle v_1, v_2 \rangle$  that applies appropriate transitions at path inputs and sets all side-inputs to the paths  $S$  to non-controlling values under  $v_2$ . The above fact will ensure that the multi-fault is both excited and sensitized by the vector  $v_2$ , i.e., the multiple fault will be testable.

## Classification of Path-delay Faults Using Stuck-at Faults

---

The converse follows from the fact that if the corresponding multiple fault in the unfolded circuit is untestable then we cannot find a suitable vector  $v_2$  that will make the set of paths MT.  $\square$

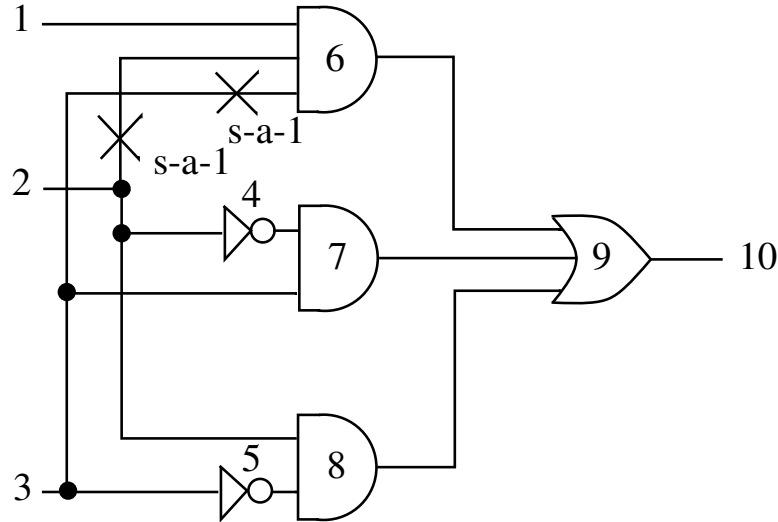


Figure 7.4: Multiply-testable paths

In Figure 7.4, the circuit shown is already unfolded as the fanouts are present only at the primary inputs. Both paths,  $\downarrow 2 - 6 - 9 - 10$  and  $\downarrow 3 - 6 - 9 - 10$ , are non-ST. However, they form an MT set of paths, testable by the vector pair  $\langle 111, 100 \rangle$ . As predicted by Theorem 7.2, the two stuck-at-1 faults shown in the figure are individually redundant but the multiple stuck-at fault constituting these single stuck-at faults is testable with the second vector  $\langle 100 \rangle$  of the above vector pair. This means if any of the two single stuck-at faults is removed, i.e., replaced by constant logic 1, the other single stuck-at fault no longer remains redundant.

Finally, if there exists a path whose corresponding stuck-at fault in the unfolded circuit is redundant and if that fault does not form a testable multi-fault with any other untestable single stuck-at fault on some input line, then it is an ST-dependent path. These are the false paths according to Gharaybeh *et al.*'s classification [42].

*Robust testability.* We now characterize the conditions on stuck-at faults in  $U$  that will ensure a path to have a robust test in  $C$ . A robust test for a path-delay fault is a test which remains valid even in the presence of delay faults along other paths. It is a non-robust test with an additional condition: if in the second vector an on-path input to some gate takes the controlling value, then all side-inputs to that gate must be stable at the non-controlling value for both vectors. For a non-robust test, it is enough to have the non-controlling value at side-inputs only in the second vector. A robust test is also a non-robust test, but the converse is not true. Hence, robust tests are essentially single-input-change (SIC) tests as for testing ST paths non-robust SIC tests are enough [41, 42].

In the circuit of Figure 7.1, the path  $\uparrow 1 - 4 - 6 - 7$  has a robust test  $\langle 00, 10 \rangle$ . The steady 0 value at primary input 2 sets all side-inputs of the path to steady non-controlling values. However, the path  $\uparrow 1 - 3 - 4 - 6 - 7$  has no robust test. It has only a non-robust test  $\langle 01, 11 \rangle$  as shown in Figure 7.5(a). Whenever this path is to be sensitized by a rising transition, a rising transition will also occur at the side-input of gate 4. But the on-path input to gate 4 will make a falling transition, i.e., towards controlling value, which will demand a steady 1 at the side-input for a robust test. So the latter path cannot have a robust test. Now consider the corresponding stuck-at faults in  $U$ . Both stuck-at faults shown in Figure 7.5 are testable as predicted by Theorem 7.1. The test for the stuck-at-0 fault on line 1 - 3 is  $\langle 11 \rangle$ ; this fault is the corresponding stuck-at fault for the path  $\uparrow 1 - 3 - 4 - 6 - 7$ . This was in fact the second vector of the non-robust test for that path. But the test of this stuck-at fault will be invalidated if the stuck-at-0 fault on line 1 - 4 is present. This is not a strange coincidence. The path  $\uparrow 1 - 4 - 6 - 7$ , whose corresponding stuck-at fault causes invalidation of the above stuck-at fault test, is actually responsible for the non-existence of the robust test for the path  $\uparrow 1 - 3 - 4 - 6 - 7$ . A

## Classification of Path-delay Faults Using Stuck-at Faults

delay fault on the former path may invalidate a non-robust test for the latter path. This is shown in Figure 7.5(c). A late rising transition on the side-input of gate 4 (due to a delay fault on the former path) will ensure that the output of gate 4 is steady 1 (instead of producing the 0 pulse). But the correct output value for gate 4 in the delay fault test of path  $\uparrow 1 - 3 - 4 - 6 - 7$  is also 1 and hence the circuit will be wrongly interpreted to be correct.

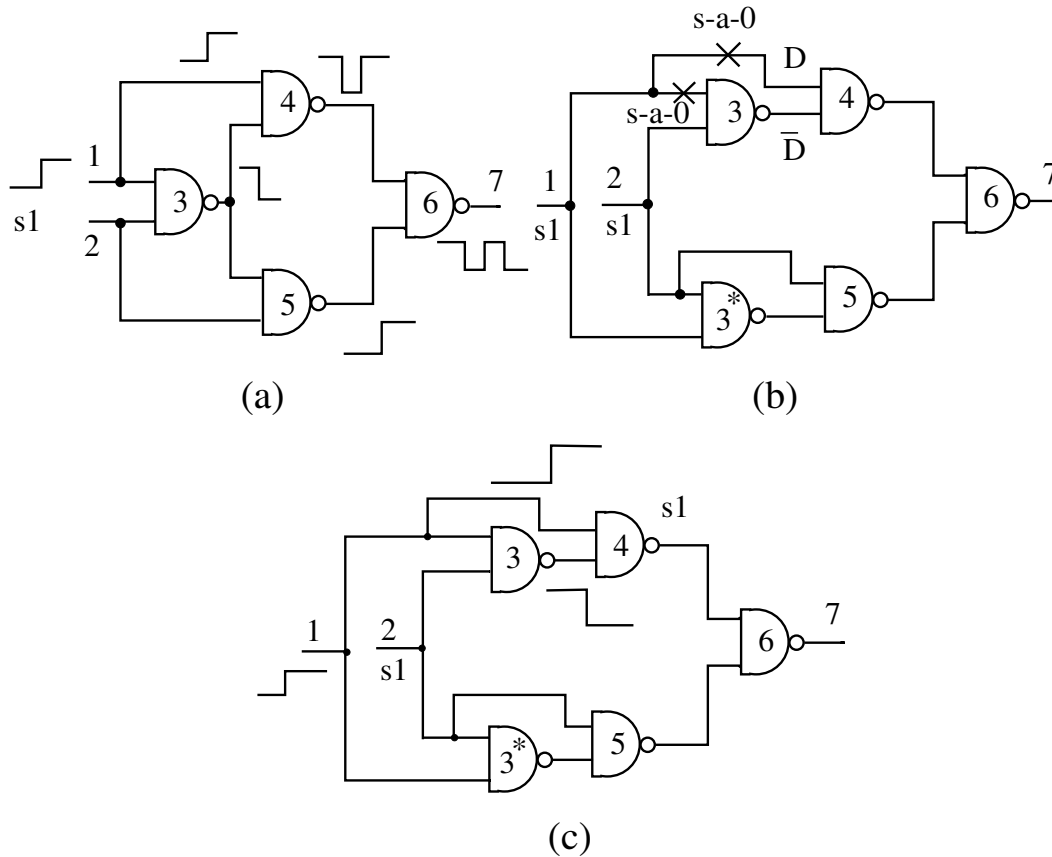


Figure 7.5: (a) Original circuit (b) Unfolded circuit (c) Limitations of a non-robust test

**Theorem 7.3** *A path-delay fault  $\uparrow P(\downarrow P)$  in a circuit is robustly testable if and only if there exists a test for its corresponding stuck-at-0 (stuck-at-1) fault in the unfolded circuit, which cannot be invalidated by the presence of any stuck-at fault in any of the other fanout branches of the same input.*

**Proof.** Let  $\langle v_1, v_2 \rangle$  be a pair of vectors, which constitutes a robust test for  $\uparrow P$ . We consider two cases:

(1) *The on-path input to some gate  $G$  on the path takes controlling value under  $v_2$ :* By definition of robust test, the side-inputs to  $G$  will be stable at non-controlling value on application of both vectors  $v_1$  and  $v_2$ . So irrespective of the logic at the input of this path, the side-inputs are stable. Hence any side path  $P'$  originating from the same primary input as path  $P$  and terminating at any side-input of  $G$  will not be sensitized by this vector pair. This will ensure that if we test the stuck-at fault corresponding to the path  $\uparrow P$  by the vector  $v_2$ , any stuck-at fault on the input branch of path  $P'$  cannot invalidate the above test.

(2) *The on-path input to some gate  $G$  takes non-controlling value under  $v_2$ :* By definition, the side-inputs must assume non-controlling value under  $v_2$ . On applying  $v_1$ , if they take non-controlling values, we have nothing to prove. However, if any of them takes the controlling value, the proof follows easily. If we test the stuck-at fault of our interest by using the vector  $v_2$ , the good value at the on-path input to gate  $G$  will be non-controlling and obviously the bad value will be controlling. A side-path  $P'$  ending at a side-input of that gate  $G$ , will also assume non-controlling value at the side-input under  $v_2$  if the circuit is good. But if the stuck-at fault corresponding to path  $P'$  is present in the unfolded circuit, the side-input to gate  $G$  will assume a controlling value (actually it will latch on to the value, that it takes under  $v_1$ ). Hence, the presence of this unwanted stuck-at fault corresponding to path  $P'$  will flag an error as it will force the bad value to pass through the gate  $G$ . Thus the original test will not be invalidated (at least a bad circuit will not be wrongly flagged as good).

It is enough to consider the stuck-at faults present only on the other fanout branches of the primary input of  $P$ . There always exists a SIC test if a path is

robustly testable. Hence, we need not consider invalidations caused by stuck-at faults present at other inputs of the unfolded circuit.

To prove the converse, we show that if there does not exist a robust test for a logical path  $P$ , any test derived to test its corresponding stuck-at fault can be invalidated by the presence of some other stuck-at fault in the unfolded circuit. If the path is non-robustly untestable, then the corresponding stuck-at fault is untestable. However, if a robust test does not exist but a non-robust test exists, then there must be a logical path  $P'$  in the original circuit starting at the same input which, when late, can invalidate the non-robust test for  $P$ . This path  $P'$  cannot join  $P$  at a gate where the path  $P$  makes a transition towards the non-controlling value under the test vector pair. In that case, both paths will be making transitions towards the non-controlling value and the slower path will actually cause the output transition giving no chance of invalidating the non-robust delay fault test. But if it joins  $P$  at a gate  $G'$  where  $P$  is making a transition towards the controlling value, then the path  $P'$ , when late, can hold the controlling value till the time  $P$  makes a transition thus removing the pulse from the output of gate  $G'$ . In this way, it can invalidate the delay fault test. If we consider the vector  $v_2$  to be the test vector for the corresponding stuck-at fault of  $P$ , then this test can be invalidated in the same manner by the presence of the corresponding stuck-at fault of  $P'$ .  $\square$

*Cheng and Chen's classification* [23]. In this classification there are two classes of path-delay faults: functionally sensitizable (FS) and redundant. The class FS properly includes the non-robust class. The non-robust class is exactly same as the ST class and so no further characterization is required. Instead we define a new class Exclusive Functional Sensitizable (EFS) = FS  $\setminus$  ST, where  $\setminus$  stands for set difference. The EFS class properly includes MT class and contains some more paths that are false paths. In Figure 7.6 we study such a path, which is ST-dependent according to Gharaybeh *et al.* [42], but

functionally sensitizable according to Cheng and Chen [23]. The conditions for a functionally sensitizable (FS) path are as follows:

(1) If the path is making a transition towards non-controlling value at any gate input, then the side-inputs should assume non-controlling value in the second vector (same as non-robust condition).

(2) If the path is making a transition towards controlling value, it does not matter what happens at the side-inputs; here the testability condition gets relaxed.

Clearly, the above definition includes the ST paths.

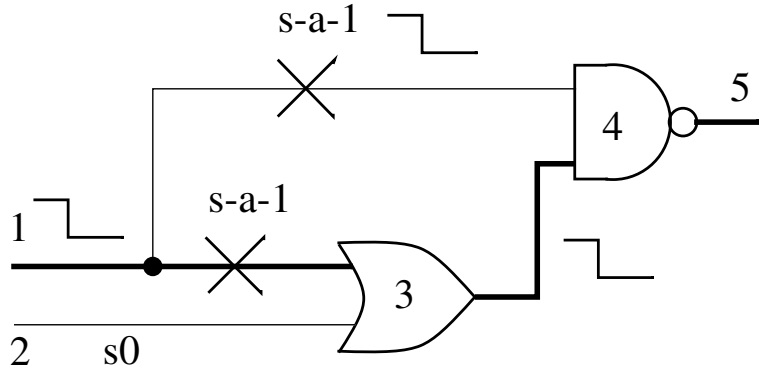


Figure 7.6: A functionally sensitized ST-dependent path

The path  $\downarrow 1 - 3 - 4 - 5$  is in FS as it follows the above definition. But it is ST-dependent on the ST path  $\downarrow 1 - 4 - 5$ , because a falling transition at the path input always brings a controlling value to the side-input of gate 4. Note that the single stuck-at-1 faults on fanout branches  $1 - 3$  and  $1 - 4$  are redundant and testable, respectively, in accordance with Theorem 7.1. However, the ST-dependent path  $\downarrow 1 - 3 - 4 - 6 - 7$  in Figure 7.3 is not in FS although the single stuck-at-1 faults at  $1 - 3$  and  $1 - 4$  are redundant and testable, respectively. So what distinguishes between these two cases? In Figure 7.3, the only vector that tests the multi-fault  $\{1 - 4: s-a-1, 1 - 3: s-a-1\}$  is 00, which is also the only vector that detects the detectable component  $1 - 4: s-a-1$  of this multi-

fault. The occurrence of the redundant component 1 – 3: s-a-1 can never be ascertained by any test vector even in the presence of other faults. However, in Figure 7.6, there exists a vector 00 which detects the multi-fault {1 – 4: s-a-1, 1 – 3: s-a-1}. Interestingly, 00 does not test the detectable component 1 – 4: s-a-1 in the absence of the redundant fault 1 – 3: s-a-1; the single fault 1 – 4: s-a-1 is detected only by 01. Thus, the test 00 detects the redundant component of the multi-fault, which becomes testable in the presence of its detectable component. So we observe that the set of EFS paths is a superset of MT paths, and hence, we can write  $EFS = MT \cup X$ . The following theorem gives a characterization for EFS paths in terms of stuck-at faults.

**Theorem 7.4** *A path  $P$  is in EFS if and only if (1) the corresponding stuck-at fault  $\alpha$  in  $U$  is redundant (here we leave aside the ST paths), and (2) There exists a stuck-at fault  $\beta$  (single or multiple) in  $U$  and a vector  $V$ , such that  $V$  detects  $\alpha \cup \beta$  but does not detect  $\beta$  alone.*

If  $\beta$  consists of only redundant fault,  $P \in MT$ , otherwise  $P$  is in class  $X$ .

**Proof.** Let  $Z$  stand for ‘Path  $P$  is in EFS’. Let  $Y$  stand for ‘ $\exists$  a stuck-at-fault  $\beta$  (single or multiple) in  $U$  and a vector  $V$ , such that  $V$  detects  $\alpha \cup \beta$  but does not detect  $\beta$  alone’.

We have to prove that  $Z \Leftrightarrow Y$ . Now  $Y \Rightarrow Z$  is equivalent to  $\bar{Z} \Rightarrow \bar{Y}$ . We use the above equivalence in the converse proof. Let  $P$  be a path that is not in FS, i.e., it is in the redundant class according to Cheng and Chen’s classification. We now consider the 8 possible combinations depending upon the direction of signal transition for both on-path and off-path inputs. For simplicity and without loss of generalization we consider only two-input gates. An  $n$ -input AND (OR, XOR) gate may be converted into a chain of  $n - 1$  2-input gates of the same type and an  $n$ -input NAND (NOR, XNOR) gate

## Classification of Path-delay Faults Using Stuck-at Faults

---

is equivalent to a chain of  $n - 1$  AND (OR, XOR) gates with the last one having a bubble at its output (See Figure 7.7 for example). This conversion does not alter the topology of the on-paths and off-paths. The formal proof is intuitively obvious and hence we omit it here.

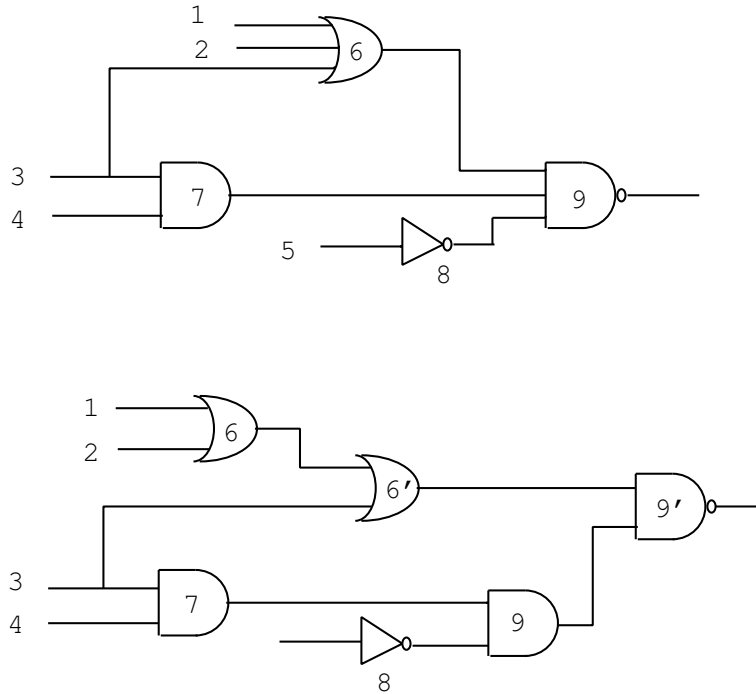


Figure 7.7: A typical circuit and its two-input equivalent

We now explain how the above configurations in Table 7.1 affect testability of a path. We consider the condition of every gate on the path under all input vector combinations. If for a path  $P_1$  there exists a vector pair for which every gate on that path has one of the configurations 1, 4, 5, and 8, then  $P_1$  belongs to the class of non-robustly testable paths. This follows easily from the definition of non-robustly testable paths. Suppose for path  $P_2$  such a vector does not exist, then it is not non-robustly testable but may be in FS or may be redundant. So for  $P_2$ , under every possible vector pair, in some gate on  $P_2$ , one of the configurations 2, 3, 6, or 7 occurs. If there exists a vector pair for

## Classification of Path-delay Faults Using Stuck-at Faults

---

Table 7.1: Possible transitions

Case	On-path input	Off-path input
1	transition	steady noncontrolling
2	towards	steady controlling
3	non-controlling	non-cont. to cont.
4	value	cont. to non-cont.
5	transition	steady noncontrolling
6	towards	steady controlling
7	controlling	non-cont. to cont.
8	value	cont. to non-cont.

which that occurrence is limited to configuration 6 and 7 only, then  $P_2$  belongs to FS. But if for a path, under every vector pair, at some gate on it, either case 2 or 3 occurs, it belongs to the redundant class. All these assertions directly follow from the definitions of different classifications of Cheng and Chen. Next, we show that if either 2 or 3 is true at any gate for any path  $P$ , then the proposition  $Y$  does not hold.

Let the untestability of path  $P$  be due to occurrence of case 2. Then for every vector  $V$ , at some gate input, the side-input remains at steady controlling value or some gate will receive complementary error values  $(D, \overline{D})$  at its two inputs. It immediately follows that the stuck-at fault at the input of the on-path  $P$  can never be tested; in other words a representative  $\beta$  cannot be found.

In case 3, the on-path and off-path signals will be going towards opposite directions. Hence, a definite transition at the output is not possible. In the best case, we can observe a pulse, but the second edge of the pulse will be determined by the off-path input as it is going towards controlling value. In the corresponding  $U$  we will have complementary error values  $(D, \overline{D})$  feeding

that gate which will make it impossible to test the fault even with any other combination of faults.

To show that  $Z \Rightarrow Y$  we consider Table 7.1 again. We have to prove that if for a vector  $v$ , the gate configurations are limited to the cases 1, 4, 5, 6, 7, or 8, then  $Y$  holds. Cases 1, 4, 5, and 8 fall under ST (non-robustly testable) category, and hence need not be considered. Thus, we need to show that for cases 6 or 7, proposition  $Y$  holds.

First, let us consider that for a path  $P$  and for every possible input vector, at least one gate  $G$  assumes the configuration 7. Further, there exists one vector for which other gates have the configurations 1, 4, 5, or 8, i.e., 6 does not occur. Thus at the gate  $G$ , both the on-path and off-path take controlling value under the second vector. If we now consider the two corresponding stuck-at faults in  $U$  for the on-path and the off-path at gate  $G$ , then the combination of these two faults will be testable. If  $G$  is the only gate with configuration 7, then all other gates will have non-controlling values at off-path inputs (1, 4, 5, or 8) in the presence of the second vector. Hence, the combination multi-fault can be propagated to some output. We thus have a representative fault  $\beta$ , which is the corresponding stuck-at fault of the off-path input. The fault  $\beta$  however, cannot be tested alone as the concerned on-path will act as off-path at gate  $G$ , and hold it to the controlling value. We show this on a typical example used by Cheng et al. [22]. In Figure 7.8 the path  $\downarrow a - x - y - z - w$  is functionally sensitizable and the gate with output  $z$  belongs to configuration 7. It is easy to observe that  $\alpha$  is the s-a-1 fault at the fanout branch of  $a$ , and  $\beta$  is the s-a-0 fault at the fanout branch of  $c$ . The multi-fault  $(\alpha, \beta)$  will be tested by the vector  $\langle 0, 1, 1 \rangle$  for inputs  $a$ ,  $b$ , and  $c$ . This vector cannot test  $\beta$  alone as  $y$  will be set to logic 0 in the absence of the stuck-at fault  $\alpha$ .

In a general case having more gates on path  $P$  with configuration 7, we construct the multi-fault by considering the corresponding stuck-at faults in  $U$

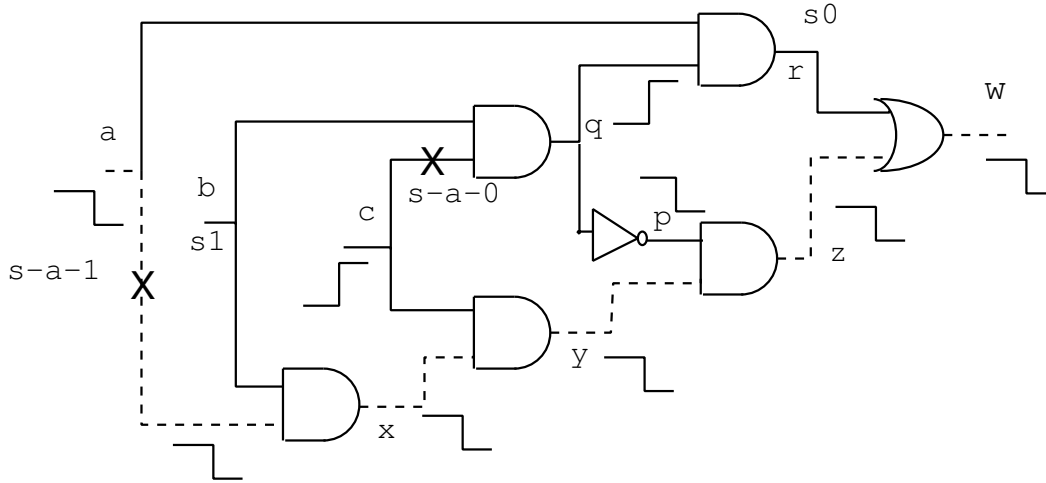


Figure 7.8: A typical example of FS path

for the on-path and all the off-paths that end at the gates with configuration 7.

Finally we consider the case 6. We assume that for every vector pair somewhere the gate configuration occurs as in the case 6. First let us assume that there is one such vector  $V$  for which configuration 6 occurs at only one gate. Let the steady controlling value at the off-path input be 0. Then the representative fault  $\beta$  is the s-a-1 fault at the off-path input (see Figure 7.9). The second vector of  $V$  will be the required test for the multi-fault  $(\alpha, \beta)$ . Note that  $\beta$  itself is unexcitable. As in the second vector, the on-path goes towards the controlling value, it cannot detect  $\beta$  alone. If configuration 6 occurs at multiple places, the multi-fault may be constructed by taking all steady controlling values. □

*Sivaraman and Strojwas' classification* [114]. This has three different classes: primitive SPDF (same as ST), primitive MPDF, and primitive-dependent. The primitive MPDF set is a proper subset of MT class, which excludes some of the false paths by defining this set carefully.

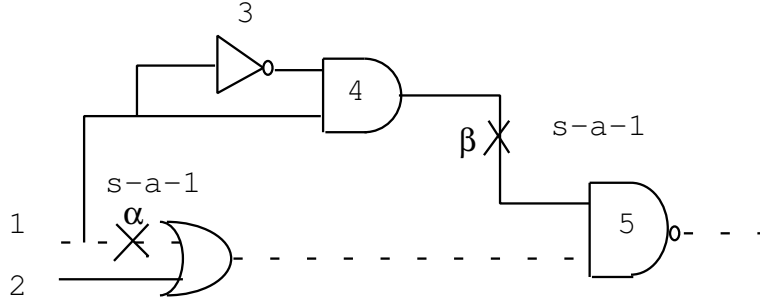


Figure 7.9: An example for Case 6

**Definition 7.5** A path-delay fault  $F$  is said to be primitive if

- 1) it is sensitizable and
- 2) none of its proper sub-faults is sensitizable [58].

The delay fault  $F$  may constitute a single path or multiple paths. The only difference between MT path and primitive MPDF is that for a set of paths  $M$  which is MT, none of the constituent paths is ST but a proper subset  $P$  of  $M$  may be as well MT. In that case the set  $M$  may be dependent on  $P$  and need not be tested separately for delay. The set,  $MT \setminus \text{primitive-MPDF} = \text{primitive-dependent} \setminus \text{ST-dependent}$ , may be termed as MT-dependent. We modify the Theorem 7.2 slightly to characterize the primitive MPDF set of paths.

**Theorem 7.5** A set of logical paths  $S$  in a circuit forms a primitive MPDF set if and only if in the unfolded circuit the multiple stuck-at fault  $M$  constituting the single stuck-at faults corresponding to the paths in  $S$  is testable. Any proper subset of  $M$  is redundant.

**Proof.** Same as Theorem 7.2 with minimality consideration on the set of faults. □

Lam *et al.* proved a theorem (Theorem 4.2 [66]) concerning robust dependent (RD) paths, which is exactly complementary to the above theorem. By

that theorem, a set of paths belongs to the RD set if and only if the multiple stuck-at fault at the input lines of the corresponding paths in the leaf-dag is redundant. A natural conclusion from this observation is that the two sets, primitive-dependent and RD, are same. In other words, in Figure 7.2, the line separating the sets non-RD and RD should be vertically aligned with the one separating the primitive MPDF and the primitive-dependent set.

### 7.4 Conclusion

We have presented a unified approach to characterize different classifications of path-delay faults in terms of stuck-at faults. Various classification schemes like ST, MT, FS, robust, RD, primitive MPDF, primitive-dependent, have been properly characterized. From a theoretical viewpoint, these characterizations make the understanding of path-delay faults much easier than before, since stuck-at fault testability is a well-studied domain for the testing community. The correlations obtained are tight and establish two-way implications. Hence, all types of path-delay faults can be re-classified using stuck-at faults alone. This work will possibly help in analyzing delay faults better, and in turn may offer an efficient tool for timing analysis in high performance circuits. In fact, timing faults due to power droop may be tested by a suitable test set that is generally based on a stuck-at fault model [97].

# Chapter 8

## Conclusions and Future Works

This thesis, primarily contributes in three emerging areas of physical design – layout-driven floorplan partitioning and routing, thermal problems in hot chips, and analysis of path-delay faults.

In layout-driven floorplan partitioning (*Chapter 2*), first an area-partitioning scheme has been proposed to facilitate global routing and repeater placement issues. The problem is to partition the floorplan hierarchically using monotonically increasing staircase cuts (MSC), so that the number of distinct nets crossing the cut is minimized, at each stage of the hierarchy [86]. A novel graph-based polynomial-time algorithm has been proposed to solve this problem. However, the two partitions obtained by this scheme are sometimes quite unbalanced in size. To take care of that, the problem has been further refined to create the partitions within a specified balance factor acceptable to the user. Several results have been proved regarding the computational hardness of the latter problem, and finally a max-flow-min-cut based heuristic has been proposed to perform the hierarchical partitioning of the floorplan. Experimental results on benchmark floorplans establishes the efficacy of our approach.

In *Chapter 3*, we investigate on a new routing scheme called topological

## Conclusions and Future Works

---

routing that tries to route all the nets simultaneously. In the traditional approach, routing is done net-by-net. So it becomes very difficult to control the congestion of routing areas, which renders the repeater placement problem very hard to handle. However, in topological routing, the routing order does not play any substantial role in determining the routability of any net. The novel method proposed in this thesis has produced very encouraging results in routing randomly generated testcases [12]. We plan to solve the associated layer assignment problem of the global routing paths and study the effect of this new routing scheme on the number of layers required compared to the traditional routers.

In *Chapter 4*, we consider a scheme for global routing. The problem is being modeled as minimizing the number of intersections among multiple Steiner trees, and an efficient heuristic is designed [90, 92]. It can help in reducing the number of expensive vias and hence may ultimately help in improving the yield of a chip as well as its timing behavior.

In the area of thermal analysis (*Chapter 5*), we have proposed a geometric model to evaluate the cumulative thermal power at any point of the chip floor. This is followed by a discrete version of the model that works in the grid domain. Then a simulation based approach is presented that identifies the hot spots and zones on a chip floor using the above models. Contrary to natural intuition, it is found that some non-source points on the chip floor can become more heated than heat-generating source points due to cumulative heating effect of nearby sources [88]. Finally, to have an overall saving in simulation time, a hybrid methodology is proposed that first uses a Voronoi diagram based technique to quickly identify the hot zones, and then uses simulation only in some focussed zones of the chip [94]. Further, to dilate the hot zones, a geometric technique is presented that will not change the floorplan adjacency relations significantly. In future, we want to tackle the time-varying sources by

augmenting our existing geometric model. We have a plan to write a simulation tool that will be able to discard superfluous data intelligently and report the time and space windows in a compact manner, when the heat content of a particular zone of the chip may cross the critical limit.

In *Chapter 6*, we introduce a new concept of *density* in an ensemble of points on a 2D floor, and prove several geometric results that facilitate the identification of regions with maximum/minimum density of source points on the chip floor. In the light of these geometric results, the algorithms required to identify the most dense (hot) zone, as well as the most rarefied (cool) zone turn out to be quite simple. The concept of density has then been compared with an existing concept of discrepancy and it is shown that there appears to be some distinct advantage in applying the concept of density over discrepancy in thermal analysis of hot spots in a chip [89]. In the theoretical side, we want to extend the concept of density in the higher dimensions and in the practical side we want to investigate how an algorithm for an alternative placement scheme can be proposed based on the concept of maximum and minimum density.

Finally, in *Chapter 7*, a very detailed theory is given how to model a path-delay fault in terms of an equivalent stuck-at fault(s) in a modified circuit derived from the original circuit. There exist several classification schemes in the path-delay fault domain. A uniform methodology is presented for finding an equivalent stuck-at fault for each of these existing classifications [84, 87]. As the stuck-at fault domain is much better understood compared to the delay fault domain, this work will possibly provide a new direction in analyzing path-delay faults, and may offer an efficient tool for timing analysis in high performance circuits.

# Bibliography

- [1] S. N. Adya and I. L. Markov, “Fixed-outline floorplanning: Enabling hierarchical design”, *IEEE Trans. VLSI Systems*, Vol. 11(6), pp. 1120-1135, 2003.
- [2] S. N. Adya, S. Chaturvedi, J. A. Roy, D. A. Papa and I. L. Markov, “Unification of partitioning, floorplanning and placement”, *Proc. ACM/IEEE Int. Conf. on Computer Aided Design (ICCAD)*, pp. 550-557, 2004.
- [3] A. Aggrawal and S. Suri, “Fast algorithms for computing the largest empty Rectangle”, *Proc. of the SOCG*, pp. 278-290, 1987.
- [4] A. Aggrawal and M. Klawe, “Applications of generalized matrix searching to geometric algorithms”, *Discrete Applied Maths.*, Vol. 27, pp. 3-23, 1990.
- [5] R. C. Aitken, “Nanometer technology effects on fault models for IC testing”, *IEEE Trans. Computers*, Vol. 48, pp. 46-51, 1999.
- [6] J. R. Alexander, J. Beck and W. W. L. Chen, “Geometric discrepancy theory and uniform distribution”, in *Handbook of Discrete and Computational Geom.*, 2nd edition, CRC Press, pp. 185-207, 1997.
- [7] F. Aurenhammer and H. Edelsbrunner, “An optimal algorithm for constructing the weighted voronoi diagram in the plane”, *Pattern Recognition*, Vol. 17, pp. 251-257, 1984.

- [8] F. Aurenhammer, "Power diagrams: Properties, algorithms and applications", *SIAM Journal of Computing*, Vol. 16, pp. 78-96, 1987.
- [9] F. Aurenhammer, "Voronoi Diagrams: A survey of a fundamental geometric data structure", *ACM Computing Survey*, Vol. 23, pp. 345-405, 1991.
- [10] F. Balasa, S. C. Maruvada and K. Krishnamoorthy, "On the exploration of the solution space in analog placement with symmetry constraints", *IEEE Trans. CAD*, Vol. 23(2), pp. 177-191, 2004.
- [11] M. D. Berg, M. V. Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 1997.
- [12] S. Bhunia, S. Majumder, S. Sur-Kolay, A. Sircar and B. B. Bhattacharya, "Topological routing amidst polygonal obstacles", *Proc. of the 13th Int. Conf. on VLSI Design*, pp. 274-279, 2000.
- [13] R. R. Brooks and S. S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*, Prentice-Hall, 1997.
- [14] S. Burman, H. Chen and N. Sherwani, "Improved global routing using  $\delta$ -geometry", *Proc. of 29th Annual Allerton Conference on Communication, Computing, and Control*, October 1991.
- [15] M. J. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits*, Kluwer Academic Publishers, Boston, 2000.
- [16] G. Calinescu, A. Dumitrescu and P. Wan, "Separating points by axis-parallel lines", *Proceedings of the CCCG*, pp. 7-10, 2004.

## BIBLIOGRAPHY

---

- [17] H. H. Chan and I. L. Markov, "Practical slicing and non-slicing block-packing without simulated annealing", *Proc. ACM/IEEE Great Lake Symp. on VLSI*, pp. 282-287, 2004.
- [18] H. H. Chan, S. N. Adya and I. L. Markov, "Are floorplan representations important in digital design?", *Proc. ISPD*, pp. 129-136, 2005.
- [19] Y. C. Chang, Y. W. Chang, G. M. Wu and S. W. Wu, " $B^*$ -trees: A new representation for nonslicing floorplans", *Proc. 37th ACM/IEEE Design Automation Conference*, pp. 458-463, 2000.
- [20] B. M. Chazelle, R. L. Drysdale III and D. T. Lee, "Computing the largest empty rectangle", *SIAM Journal of Computing*, Vol. 15(1), pp. 300-315, 1986.
- [21] B. Chazelle and H. Edelsbrunner, "An improved Algorithm for constructing kth-order Voronoi diagram in the Plane", *IEEE Transactions on Computers*, Vol. C-36, pp. 1349-1354, 1987.
- [22] K.-T. Cheng and H. C. Chen, "Delay testing for non-robust untestable circuits", *Proc. International Test Conf.*, pp. 954-961, 1993.
- [23] K.-T. Cheng and H.-C. Chen, "Classification and identification of non-robust untestable path delay faults", *IEEE Trans. CAD*, Vol. 15, pp. 845-853, 1996.
- [24] Y. K. Cheng and S. M. Kang, "An efficient method for hot-spot identification in ULSI circuits", *Proc. ACM/IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 124-127, 1999.
- [25] C. N. Chu and D. F. Wong, "A matrix synthesis approach to thermal placement", *Proc. of Int. Symp. on Physical Design*, pp. 163-168, 1997.

- [26] C. C. N. Chu and D. F. Wong, "Closed form solution to simultaneous buffer insertion/sizing and wire sizing", *Proc. Int. Symp. on Physical Design*, pp. 192-197, 1997.
- [27] J. Cong, L. He, C.-K. Koh and P. H. Madden, "Performance optimization of VLSI interconnect layout", *Integration, the VLSI Journal*, Vol. 21, pp. 1-94, 1996.
- [28] J. Cong, "Challenges and opportunities for design innovations in nanometer technologies", *Frontier in Semiconductor Research: A Collection of SRC Working Papers*, Semiconductor Research Corporation, [http://www.src.org/prg\\_mgmt/frontier.dgw](http://www.src.org/prg_mgmt/frontier.dgw), 1997.
- [29] J. Cong, T. Kong and D. Z. Pan, "Buffer block planning for interconnect-driven floorplanning", *Proc. ACM/IEEE Int. Conf. on Computer Aided Design (ICCAD)*, pp. 358-363, 1999.
- [30] J. Cong, T. Kong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning", *Proc. Asia South Pacific Design Automation Conference*, pp. 97-100, 1999.
- [31] T. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
- [32] W. W. Dai, R. Kong and J. Jue, "Rubber-band routing and dynamic data representation", *Proc. 1990 International Conference on Computer-Aided Design*, pp. 52-55, 1990.
- [33] W. W. Dai, T. Dayan and D. Staepfleare, "Topological routing in SURF: Generating a rubber-band sketch", *Proc. 28th Design Automation Conference*, pp. 39-41, 1991.

## BIBLIOGRAPHY

---

- [34] S. Das, S. Sur-Kolay and B. B. Bhattacharya, "Manhattan-diagonal routing in channels and switchboxes", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol 9(1), pp. 75-104, 2004.
- [35] P. S. Dasgupta, A. K. Sen, S. C. Nandy and B. B. Bhattacharya, "Searching networks with unrestricted edge costs", *IEEE Transactions on Systems, Man and Cybernetics: Part A*, Vol. 31, pp. 497-507, November 2001.
- [36] P. S. Dasgupta, P. Pan, S. C. Nandy and B. B. Bhattacharya, "Monotone bipartitioning problem in a planar point set with applications to VLSI", *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 7, no. 2, pp. 231-248, 2002.
- [37] A. Dumitrescu and J. Pach, "Partitioning colored point sets into monochromatic parts", *WADS 2001, LNCS*, Vol. 2125, pp. 264-275, 2001.
- [38] Y. Feng, D. P. Mehta and H. Yang, "Constrained modern floorplanning", *Proc. ISPD*, pp. 128-134, 2003.
- [39] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.
- [40] D. R. Gaur, T. Ibaraki and R. Krishnamurti, "Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem", *Journal of Algorithms*, Vol. 43, pp. 138-152, 2002.
- [41] M. A. Gharaybeh, "Testing for timing correctness of high-speed VLSI circuits", *Dissertation*, ECE Dept., Rutgers University, Oct 1996.
- [42] M. A. Gharaybeh, M. L. Bushnell and V. D. Agrawal, "Classification and test generation for path-delay faults using single stuck-fault tests", *J. Electronic Testing: Theory and Applications*, Vol. 11, pp. 55-67, 1997.

- [43] M. A. Gharaybeh, M. L. Bushnell and V. D. Agrawal, "False-path removal using delay fault simulation", *Proc. 7th IEEE Asian Test Symp.*, pp. 82-87, 1998.
- [44] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem", *Journal of the ACM*, Vol. 35(4), pp. 921-940 1988.
- [45] M. Guruswamy and D. F. Wong, "Channel routing order for building-block layout with rectilinear modules", *Proc. ACM/IEEE Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 184-187, 1988.
- [46] S. Haruyama, D. F. Wong and D. S. Fusell, "Topological channel routing", *IEEE Trans. on CAD*, Vol. 10(10), pp. 1117-1119, 1992.
- [47] R. Hassin and N. Megiddo, "Approximation algorithms for hitting objects with straight lines", *Discrete Applied Math.*, Vol. 30, pp. 29-42, 1991.
- [48] K. Heragu, J. H. Patel and V. D. Agrawal, "Fast identification of untestable delay faults using implications", *Proc. ACM/IEEE Int. Conf. on Computer Aided Design (ICCAD)*, pp. 642-647, 1997.
- [49] J. M. Ho, G. Vijayan and C. K. Wong, "A new approach to the rectilinear Steiner tree problem", *IEEE Transactions on Computer-Aided Design*, Vol. 9(2), pp. 185-193, 1985.
- [50] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C.-K. Cheng and J. Gu, "Corner block list: An effective and efficient topological representation of nonslicing floorplan", *Proc. ACM/IEEE Int. Conf. on Computer Aided Design (ICCAD)*, pp. 8-12, 2000.
- [51] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan and S. Velusamy, "Compact thermal modeling for temperature-aware

## BIBLIOGRAPHY

---

- design”, *Proc. 41st ACM/IEEE Design Automation Conference*, pp. 878883, 2004.
- [52] W. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T.Theocharides and M. J. Irwin, “Thermal-aware floorplanning using genetic algorithms”, *Proc. Sixth International Symposium on Quality of Electronic Design (ISQED’05)*, 2005.
- [53] F. K. Hwang, “On steiner minimal trees with rectilinear distance”, *SIAM Journal of Applied Mathematics* Vol. 30(1), pp. 104-114, 1976.
- [54] F. K. Hwang, “An  $O(n \log n)$  algorithm for suboptimal rectilinear Steiner trees”, *IEEE Transactions on Circuits and Systems*, Vol. 26(1), pp. 75-77, 1979.
- [55] A. Jagannathan, S. W. Hur and J. Lillis, “A fast algorithm for context-aware buffer insertion”, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 7(1), pp. 173-188, 2002.
- [56] S. M. Kang, “On-chip thermal engineering for peta-scale integration”, *Proc. ISPD*, p. 76, 2002.
- [57] M. Kano, C. Merino and J. Urrutia, “On spanning trees and cycles of multicolored point sets with few intersections”, *Information Processing Letters*, Manuscript.
- [58] W. Ke and P. R. Menon, “Synthesis of delay-verifiable combinational circuits”, *IEEE Trans. on Computers*, Vol. 44, pp. 213-222, 1995.
- [59] V. King, S. Rao and R. E. Tarjan, “A faster deterministic maxflow algorithm”, *Journal of Algorithms*, Vol 17(3), pp. 447-474, 1995.

- [60] R. Klein, "Abstract Voronoi diagrams and their applications", *Computational Geometry and its Applications: LNCS*, Vol. 333, pp. 148-157, Springer-Verlag, 1988.
- [61] F. Koushanfar, S. Slijepcevic, M. Potkonjak and A. Sangiovanni-Vincentelli, "Error-tolerant multi-modal sensor fusion", *IEEE CAS Workshop on Wireless Communication and Networking*, Pasadena 2002.
- [62] A. Krisnamachary and J. A. Abraham, "Effects of multicycle sensitization on delay tests", *Proc. International Conf. VLSI Design*, pp. 137-142, 2003.
- [63] A. Krstić and K.-T. Cheng, *Delay Fault Testing for VLSI Circuits*, Boston: Kluwer Academic Publishers, 1998.
- [64] M. Lai and D. Wong, "Slicing tree is a complete floorplan representation", *Proc. Design Automation and Test Europe (DATE)*, pp. 228-232, 2001.
- [65] S. Lakshiminarayanan, P. J. Wright and J. Pallinti, "Electrical characterization of the copper CMP process and derivation of metal layout rules", *IEEE Trans. on Semiconductor Manu.*, Vol. 16(4), pp. 668-676, 2003.
- [66] W. K. Lam, A. Saldanha, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Delay fault coverage, test set size, and performance trade-offs", *IEEE Trans. CAD*, Vol. 14, pp. 32-44, 1995.
- [67] D. T. Lee, "Two-dimensional Voronoi diagrams in the  $L - p$ -metric", *JACM*, Vol. 27, pp. 604-618, 1980.
- [68] H.-C. Lee, Y.-W. Chang, J.-M. Hsu and H. H. Yang, "Multilevel floorplanning/placement for large-scale modules using  $B^*$ -trees", *Proc. 40th ACM/IEEE Design Automation Conference*, pp. 812-817, 2003.

## BIBLIOGRAPHY

---

- [69] C. E. Leiserson and F. M. Maley, "Algorithms for routing and testing routability of planar VLSI layouts", *Proc. 17th Annual ACM Symposium on Theory of Computing*, pp. 69-78, 1985.
- [70] H. C. Liang, C. L. Lee and J. E. Chen, "Identifying untestable faults in sequential circuits", *IEEE Design and Test of Computers*, Vol. 12(3), pp. 12-23, 1995.
- [71] K. F. Liao, M. Sarrafzadeh and C. K. Wong, "Single-layer global routing", *IEEE Trans. on CAD*, Vol. 13(1), pp. 38-47, 1994.
- [72] A. Lim, S. Sahni and V. Thanvantri, "A fast algorithm to test planar topological routability", *Proc. International Conference on VLSI Design*, pp. 8-12, 1995.
- [73] C. J. Lin and S. M. Reddy, "On delay fault testing in logic circuits", *IEEE Trans. on CAD*, Vol.6, pp. 694-703, 1987.
- [74] J. Lin and Y. Chang, "TCG: A transitive closure graph based representation for nonslicing floorplans", *Proc. 38th ACM/IEEE Design Automation Conference*, pp. 764-769, 2001.
- [75] J.-M. Lin, Y.-W. Chang and S.-P. Lin, "Corner sequence – A P-admissible floorplan representation with a worst case linear-time packing scheme", *IEEE Trans. on VLSI Systems*, Vol. 11(4), pp. 679-686, 2003.
- [76] H. Liu and D. F. Wong, "Network flow based multi-way partitioning with area and pin constraint", *IEEE Trans. on Computer-Aided Design*, Vol. 17, pp. 50-59, 1998.
- [77] X. Liu, M. S. Hsiao, S. Chakravarty and P.J. Thadikaran, "Efficient transition fault ATPG algorithms based on stuck-at test vectors", *Journal of Electronic Testing: Theory and Applications*, Vol. 19(4), pp. 437-445, 2003.

- [78] B. Lu, Y. Xu, B. Zhu and D. Du, "On a minimum linear classification problem", *Journal of Global Optimization*, Vol. 26(4), pp. 435-441, 2003.
- [79] A. K. Majhi, J. Jacob, L. M. Patnaik and V. D. Agrawal, "On test coverage of path delay faults", *Proc. 9th International Conf. VLSI Design*, pp. 418-421, 1996.
- [80] A. K. Majhi and V. D. Agrawal, "Tutorial: Delay fault models and coverage", *Proc. 11th International Conf. VLSI Design*, pp. 364-369, 1998.
- [81] S. Majumder, *Routing driven floorplan partitioning using staircase channel*, M. Tech. Dissertation Thesis, Indian Statistical Institute, July 1996.
- [82] S. Majumder, B. B. Bhattacharya and S. C. Nandy, "Partitioning VLSI floorplans by staircase channels for global routing", *Proc. of the 11th Int. Conf. on VLSI Design*, pp. 59-64, 1998.
- [83] S. Majumder, V. D. Agrawal and M. L. Bushnell, "On delay-untestable paths and stuck-fault redundancy", *Proc. 16th IEEE VLSI Test Symp.*, pp. 194-199, 1998.
- [84] S. Majumder, B. B. Bhattacharya, V. D. Agrawal and M. L. Bushnell, "A complete characterization of path delay faults through stuck-at faults", *Proc. of the 12th Int. Conf. on VLSI Design*, pp. 492-497, 1999.
- [85] S. Majumder, S. Sur-Kolay, B. B. Bhattacharya and S. C. Nandy, "Area(Number)-balanced hierarchy of staircase channels with minimum crossing nets", *Proc. Int. Symposium on Circuits and Systems (ISCAS)*, Vol. 5, pp. 395-398, 2001.
- [86] S. Majumder, S. C. Nandy and B. B. Bhattacharya, "On finding a staircase channel with minimum crossing nets in a VLSI floorplan", *Journal*

## BIBLIOGRAPHY

---

- of Circuits, Systems, and Computers (JCSC)*, Vol 13(5), pp. 1019-1038, 2004.
- [87] S. Majumder, B. B. Bhattacharya, V. D. Agrawal and M. L. Bushnell, "A new classification of path-delay fault testability in terms of stuck-at faults", *Journal of Computer Science and Technology*, Vol. 19(6), pp. 955-964, 2004.
- [88] S. Majumder, S. Sur-Kolay, S. C. Nandy, B. B. Bhattacharya and B. Chakraborty, "Hot spots and zones in a chip: A geometrician's view", *Proc. Int. Conf. VLSI Design*, pp. 691-696, 2005.
- [89] S. Majumder and B. B. Bhattacharya, "Density or discrepancy? A VLSI designer's dilemma in hot spot analysis", *Proc. of the Canadian Conference on Computational Geometry*, pp. 167-170, 2005.
- [90] S. Majumder, B. B. Bhattacharya and S. M. A. Jafri, "Reducing crossing number of multi-color rectilinear Steiner trees using monochromatic partitioning", submitted for publication, 2005.
- [91] S. Majumder, S. Sur-Kolay, B. B. Bhattacharya and S. Das, "Hierarchical partitioning of VLSI floorplans by staircases", *submitted for journal publication*, 2005.
- [92] S. Majumder, S. C. Nandy and B. B. Bhattacharya, "Separating multi-color points on a plane with fewest axis-parallel lines", *Tech. Report ISI/ACMU/VLSI - 1*, Kolkata, 2005.
- [93] S. Majumder, S. Sur-Kolay, S. Bhunia and B. B. Bhattacharya, "Topological routing for a group of nets", *Tech. Report ISI/ACMU/VLSI - 2*, Kolkata, 2005.

- [94] S. Majumder and B. B. Bhattacharya, "Solving thermal problems of hot chips using Voronoi diagrams", *Proc. of the 19th Int. Conference on VLSI Design*, January 2006 (to appear).
- [95] M. Marek-Sadowska and T. T.-K. Tarng, "Single-layer routing for VLSI: Analysis and algorithms", *IEEE Trans. on CAD*, Vol. CAD-2(4), pp. 246-259, 1983.
- [96] G. Miranda, H. P. L. Luna, G. R. Meteus and R. P. M. Ferreira, "A performance guarantee heuristic for electronic components placement problems including thermal effects", *Computers and Operations Research*, Vol. 32(11), pp. 2937-2957, 2005.
- [97] D. Mitra, S. Bhattacharjee, S. Sur-Kolay, B. B. Bhattacharya, S. T. Zachariah and S. Kundu, "Test pattern generation for power supply droop faults", *Proc. of the 19th Int. Conference on VLSI Design*, January 2006 (to appear).
- [98] H. Murata, H. Fujiyoshi, S. Nakatake and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair", *IEEE Trans. on Computer-Aided Design*, Vol. 15, pp. 1518-1524, 1996.
- [99] S. C. Nandy and B. B. Bhattacharya, "A unified algorithm for finding maximum and minimum object enclosing rectangles and cuboids", *Computers and Mathematics with Applications*, Vol. 29, pp. 45-61, 1994.
- [100] S. C. Nandy and B. B. Bhattacharya, "On finding an empty staircase polygon of largest area (width) in a planar point-set", *Computational Geometry: Theory and Applications*, Vol. 26, pp. 143-171, 2003.
- [101] W. N. Needham, C. Prunty, E. H. Yeoh, "High volume microprocessor test escapes, an analysis of defects our tests are missing", *Proc. International Test Conf.*, pp. 25-34, 1998.

## BIBLIOGRAPHY

---

- [102] T. Okamoto and J. Cong, "Interconnect layout optimization by simultaneous Steiner tree construction and buffer insertion", *Proc. ACM/SIGDA Physical Design Workshop*, pp. 1-6, 1996.
- [103] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, N.J., 1982.
- [104] I. Pomeranz and S. M. Reddy, "On the number of tests to detect all path delay faults in combinational logic circuits", *IEEE Trans. Computers*, Vol. 45, pp. 50-62, 1996.
- [105] R. Puri, L. Stok and S. Bhattacharya, "Keeping hot chips cool", *Proc. 42nd ACM/IEEE Design Automation Conference*, pp. 285-288, 2005.
- [106] F. Rafiq, M. Chrzanowska-Jeske, H. H. Yang, M. Jeske and N. Sherwani, "Integrated floorplanning with buffer/channel Insertion for bus-based designs", *IEEE Trans. on Computer-Aided-Design*, Vol. 22(6), pp. 730-741, 2003.
- [107] S. M. Sait and H. Youssef, *VLSI Physical Design Automation: Theory and Practice*, World Scientific, Singapore, 1999.
- [108] K. Sakanushi, Y. Kajitani and D. P. Mehta, "The quarter-state sequence floorplan representation", *IEEE Trans. on Circuits and Systems - I*, Vol. 50, pp 376-386, 2003.
- [109] A. Saldanha, R. Brayton and A. Sangiovanni-Vincentelli, "Equivalence of robust delay-fault and single stuck-fault test generation", *Proc. 29th Design Automation Conf.*, pp. 173-176, 1992.
- [110] P. Sarkar and C.-K. Koh, "Routability-driven repeater block planning for interconnect-centric floorplanning", *IEEE Trans. on Computer-Aided Design*, Vol. 20, pp. 660-671, 2001.

- [111] M. Segal and K. Kedem, "Enclosing k points in the smallest axis parallel rectangle", *Info. Process. Letters*, Vol. 65(2), pp. 95-99, 1998.
- [112] N. A. Sherwani, S. Bhingrade and A. Panyam, *Routing in the Third Dimension*, IEEE Press, 1995.
- [113] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Third Edition, Kluwer Academic Publishers, Boston, 1999.
- [114] M. Sivaraman and A. J. Strojwas, "Primitive path delay fault identification", *Proc. 10th International Conf. VLSI Design*, pp. 95-100, 1997.
- [115] G. L. Smith, "Model for delay faults based upon paths", *Proc. International Test Conf.*, pp. 342-349, 1985.
- [116] U. Sparmann, D. Luxenburger, K.-T. Cheng and S. M. Reddy, "Fast identification of robust dependent path delay faults", *Proc. 30th Design Automation Conf.*, pp. 119-125, 1995.
- [117] U. Sparmann and L. Koeller, "Improving path delay fault testability by path removal", *Proc. 16th IEEE VLSI Test Symp.*, pp. 200-208, 1998.
- [118] R. Sun, R. Gupta and C. L. Liu, "Congestion-balanced placement for FPGAs", *Proc. of 5th Physical Design Workshop*, pp. 163-168, 1996.
- [119] S. Sur-Kolay and B. B. Bhattacharya, "The cycle structure of channel graphs in non-sliceable floorplans and a unified algorithm for feasible routing order", *Proc. Int. Conf. on Computer Design (ICCD)*, IEEE CS Press, USA, pp. 524-529, 1991.
- [120] X. Tang, R. Tian and D. F. Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation", *Proc. Design Automation and Test Europe (DATE)*, pp. 106-111, 2000.

## BIBLIOGRAPHY

---

- [121] X. Tang and D. F. Wong, "FAST-SP: A fast algorithm for block placement based on sequence pair", *Proc. Asia South Pacific Design Automation Conference*, pp. 521-526, 2001.
- [122] E. Tardos, "A strongly polynomial algorithm to solve combinatorial linear programs", *Operations Research*, Vol. 34, pp. 250-256, 1986.
- [123] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, 1983.
- [124] R. Tekumalla and P. R. Menon, "Identifying redundant path delay faults in sequential circuits", *Proc. 9th International Conf. on VLSI Design*, pp. 406-411, 1996.
- [125] C. Tirumurti, S. Kundu, S. Sur-Kolay and Y.-S. Chang, "A modeling approach for addressing power supply switching noise related failures of integrated circuits", *Proc. Design Automation and Test Europe (DATE)*, pp. 1078-1083, 2004.
- [126] S. Tokunaga, "Intersection number of two connected geometric graphs", *Information Processing Letters*, Vol. 59, pp. 331-333, 1996.
- [127] C. H. Tsai and S. M. Kang, "Cell-level placement for improving substrate thermal distribution", *IEEE Transactions on CAD*, Vol. 19, pp. 253-265, 2000.
- [128] D. M. Warme, "A new exact algorithm for rectilinear Steiner trees", *International Symposium on Mathematical Programming*, 1997.
- [129] D. F. Wong and C. L. Liu, "A new algorithm for floorplan design", *Proc. ACM/IEEE Design Automation Conference*, pp. 101-107, 1986.

- [130] H. H. Yang and D. F. Wong, “Efficient network flow based min-cut balanced partitioning”, *IEEE Trans. Computer-Aided Design*, Vol. 15, pp. 1533–1540, Dec. 1996.
- [131] B. Yao, H. Chen, C. K. Cheng and R. Graham, “Floorplan representations: Complexity and connections”, *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, Vol. 8(1), pp. 55-80, 2003.
- [132] E. F. Y. Young, C. C. N. Chu and Z. C. Shen, “Twin binary sequences: A nonredundant representation for general nonslicing floorplan”, *IEEE Trans. CAD*, Vol. 22(4), pp. 457-469, 2003.
- [133] H. Zhou and J. Wang, “ACG – Adjacent constraint graph for general floorplans”, *Proc. Int. Conf. on Computer Design (ICCD)*, pp. 572-575, 2004.
- [134] C. Zhuang, K. Sakanushi, L. Jin and Y. Kajitani, “An enhanced Q-sequence augmented with empty-room-insertion and parenthesis trees”, *Proc. Design Automation and Test Europe (DATE)*, pp. 61-68, 2002.
- [135] “International Technology Roadmap for Semiconductors”, Nov. 2002 (<http://www.itrs.net>).